G-18 Multivariate and Regression Analysis

November 21, 2020

1 G-18 EE Climate Change Project- Multivariate and Regression Analysis (Notebook 2 of 2)

Additional Packages required to run the code:

- conda update –all
- conda install geopandas
- conda install descartes

1.1 Table of Contents

S No	Section	Topic
	5.7	Comparison of Variables of Past and Future for Bangalore

1.2 Done By: -

Name	Roll Number
Vimal Rajesh	B180336CS
Puchakayala Dheeraj Reddy	B180902CS
P Arjun	B180454CS
Alok Raj	B180411CS
Kunal Ravikumar Jagtap	B180921CS

2 For India

2.1 Loading All Districts Cleaned Datasets

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import geopandas as gpd
import warnings
warnings.filterwarnings('ignore')
```

Variable Number 1: AOD

Aerosol optical depth is a measure of the extinction of the solar beam by dust and haze. In other words, particles in the atmosphere (dust, smoke, pollution) can block sunlight by absorbing or by scattering light. AOD tells us how much direct sunlight is prevented from reaching the ground by these aerosol particles.

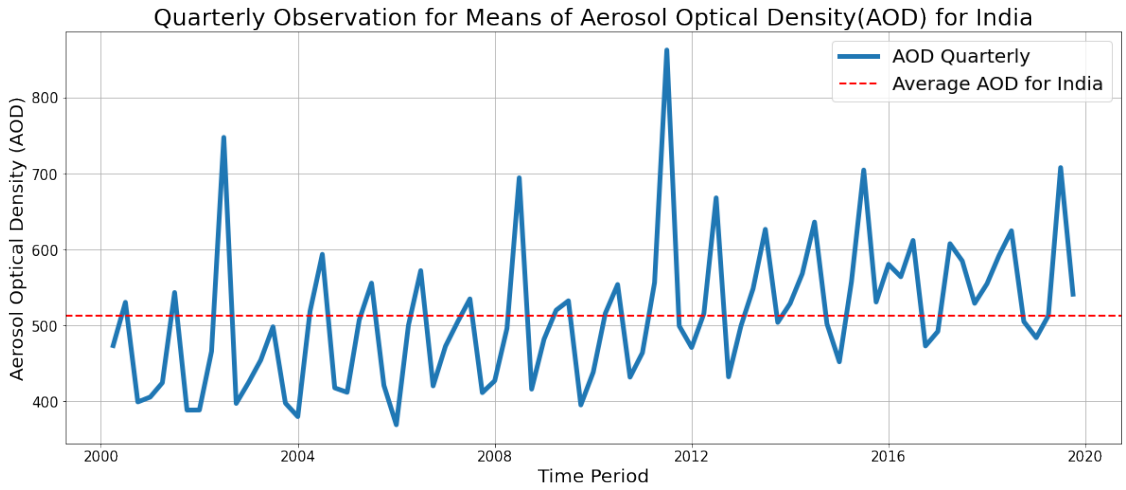
[2]:
```
aod_india = pd.read_csv("India_Final/aod.csv")
aod_india.ST_NM = aod_india.ST_NM.str.lower()
aod_india.date = pd.to_datetime(aod_india.date)
aod_india
```

[2]:

	ST_NM	date	aod
0	andaman & nicobar island	2000-04-01	216.060785
1	andaman & nicobar island	2000-07-01	317.311594
2	andaman & nicobar island	2000-10-01	341.136057
3	andaman & nicobar island	2001-01-01	192.204372
4	andaman & nicobar island	2001-04-01	244.904924
…	…	…	…
2748	west bengal	2018-10-01	655.586612
2749	west bengal	2019-01-01	856.090348
2750	west bengal	2019-04-01	751.726200

```
2751              west bengal 2019-07-01  837.340381
2752              west bengal 2019-10-01  758.595713

[2753 rows x 3 columns]
```

[3]:
```
# Visualizing by grouping into quarterly data
aod_india_date = aod_india.groupby("date").mean()
plt.figure(figsize = (20,8))
plt.plot( aod_india_date.index.tolist(),aod_india_date["aod"], linewidth = 5,␣
 ↪label = "AOD Quarterly")
plt.gca().axhline(y=aod_india_date["aod"].mean(),color='r', linestyle='--',␣
 ↪lw=2, label = "Average AOD for India")
plt.title("Quarterly Observation for Means of Aerosol Optical Density(AOD) for␣
 ↪India",fontsize = 25)
plt.xlabel("Time Period",fontsize= 20)
plt.ylabel("Aerosol Optical Density (AOD)",fontsize= 20)


plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
# plt.gca().set_yticklabels(plt.gca().get_yticklabels(),fontsize = 10)
plt.legend(fontsize = 20)
plt.grid()
```



The graph show that the mean Aerosol Optical Density(AOD) in India have increased in the past 20 years. Most of the time in 2000-2002 the value of the AOD is below the average. But this has grown to greater values. There have been large peaks in 2003 and 2011 signifing rapid increase in the AOD values.

Variable Number 2 : Evapotranspiration (ET)

It is the sum of evaporation and plant transpiration from the Earth's land and ocean surface to the

atmosphere. Evaporation accounts for the movement of water to the air from sources such as the soil, canopy interception, and water bodies.

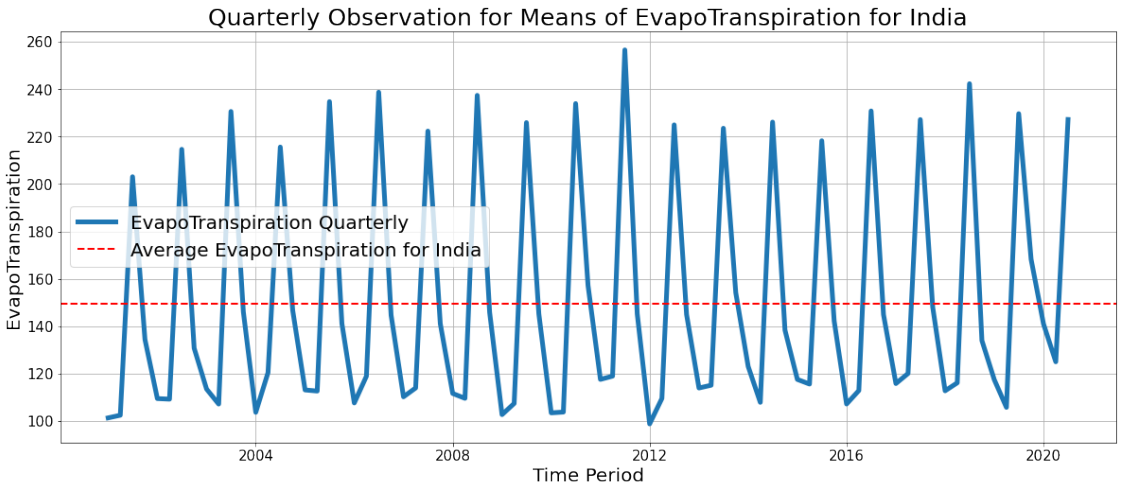
[4]:
```
evaptrans_india = pd.read_csv("India_Final/evaptrans.csv")
evaptrans_india.ST_NM = evaptrans_india.ST_NM.str.lower()
evaptrans_india.date = pd.to_datetime(evaptrans_india.date)
evaptrans_india
```

[4]:

	ST_NM	date	evaptrans
0	andaman & nicobar island	2001-01-01	361.736801
1	andaman & nicobar island	2001-04-01	304.005540
2	andaman & nicobar island	2001-07-01	235.343897
3	andaman & nicobar island	2001-10-01	272.322672
4	andaman & nicobar island	2002-01-01	350.833868
…	…	…	…
2760	west bengal	2019-07-01	222.791903
2761	west bengal	2019-10-01	176.575370
2762	west bengal	2020-01-01	112.953953
2763	west bengal	2020-04-01	137.917302
2764	west bengal	2020-07-01	224.001299

[2765 rows x 3 columns]

[5]:
```
# Visualizing by grouping into quarterly data
evaptrans_india_date = evaptrans_india.groupby("date").mean()
plt.figure(figsize = (20,8))
plt.plot( evaptrans_india_date.index.
 ↪tolist(),evaptrans_india_date["evaptrans"], linewidth = 5, label =␣
 ↪"EvapoTranspiration Quarterly")
plt.gca().axhline(y=evaptrans_india_date["evaptrans"].mean(),color='r',␣
 ↪linestyle='--', lw=2, label = "Average EvapoTranspiration for India")
plt.title("Quarterly Observation for Means of EvapoTranspiration for␣
 ↪India",fontsize = 25)
plt.xlabel("Time Period",fontsize= 20)
plt.ylabel("EvapoTranspiration",fontsize= 20)

plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
# plt.gca().set_yticklabels(plt.gca().get_yticklabels(),fontsize = 10)
plt.legend(fontsize = 20)
plt.grid()
```



The grapth shows the seasonal variation of the evapotranspiration values in India. Plotting the Quaterly values of the mean evapotranspiration, there have been seasonal variation in the Evapotranpiration.

Variable Number 3: Fire from MODIS_FIRE

The fire detection strategy is based on absolute detection of a fire (when the fire strength is sufficient to detect), and on detection relative to its background (to account for variability of the surface temperature and reflection by sunlight). The product distinguishes between fire, no fire and no observation. This information is used for monitoring the spatial and temporal distribution of fires in different ecosystems, detecting changes in fire distribution and identifying new fire frontiers, wild fires, and changes in the frequency of the fires or their relative strength.

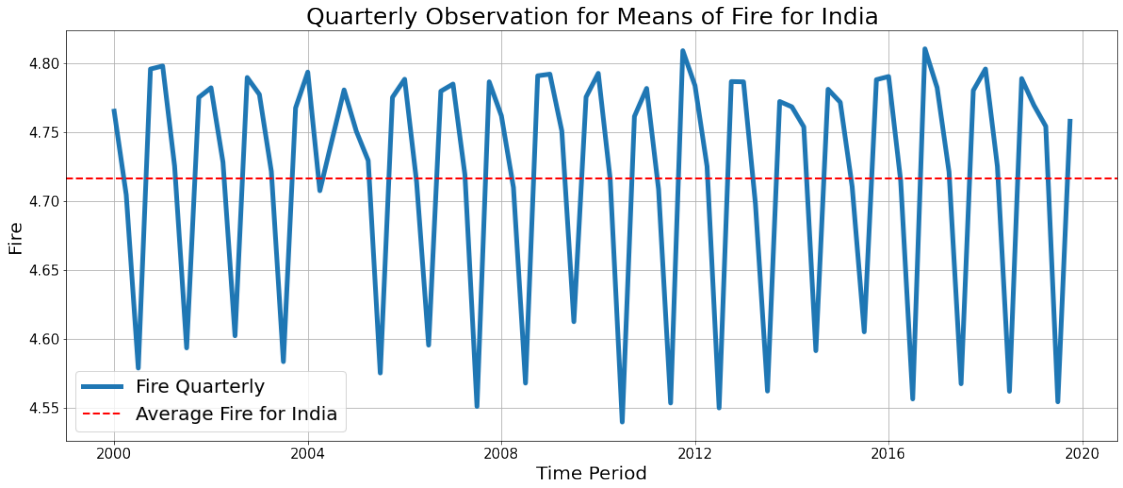
[6]:
```
fire_india= pd.read_csv("India_Final/fire.csv")
fire_india.State = fire_india.State.str.lower()
fire_india.date = pd.to_datetime(fire_india.date)
fire_india
```

[6]:

	State	date	fire
0	andaman & nicobar island	2000-01-01	4.480148
1	andaman & nicobar island	2000-04-01	4.360132
2	andaman & nicobar island	2000-07-01	4.341937
3	andaman & nicobar island	2000-10-01	4.468958
4	andaman & nicobar island	2001-01-01	4.525918
…	…	…	…
2760	west bengal	2018-10-01	4.872379
2761	west bengal	2019-01-01	4.875533
2762	west bengal	2019-04-01	4.796782
2763	west bengal	2019-07-01	4.595394
2764	west bengal	2019-10-01	4.841796

[2765 rows x 3 columns]

[7]:
```
# Visualizing by grouping into quarterly data
fire_india_date = fire_india.groupby("date").mean()
plt.figure(figsize = (20,8))
plt.plot( fire_india_date.index.tolist(),fire_india_date["fire"], linewidth =␣
 ↪5, label = "Fire Quarterly")
plt.gca().axhline(y=fire_india_date["fire"].mean(),color='r', linestyle='--',␣
 ↪lw=2, label = "Average Fire for India")
plt.title("Quarterly Observation for Means of Fire for India",fontsize = 25)
plt.xlabel("Time Period",fontsize= 20)
plt.ylabel("Fire",fontsize= 20)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
# plt.gca().set_yticklabels(plt.gca().get_yticklabels(),fontsize = 10)
plt.legend(fontsize = 20)
plt.grid()
```



The graph shows the mean values of Fires in India. The range of the values of the fire being low doesnot give a scope to interpret from the graph. But a increase in the fires in the summer of all years signifies that the number of fires increase in summer in India

Variable Number 4 : Precipitation

Precipitation is rain, snow, sleet, or hail — any kind of weather condition where something's falling from the sky. Precipitation has to do with things falling down, and not just from the sky. It's also what happens in chemical reactions when a solid settles to the bottom of a solution.

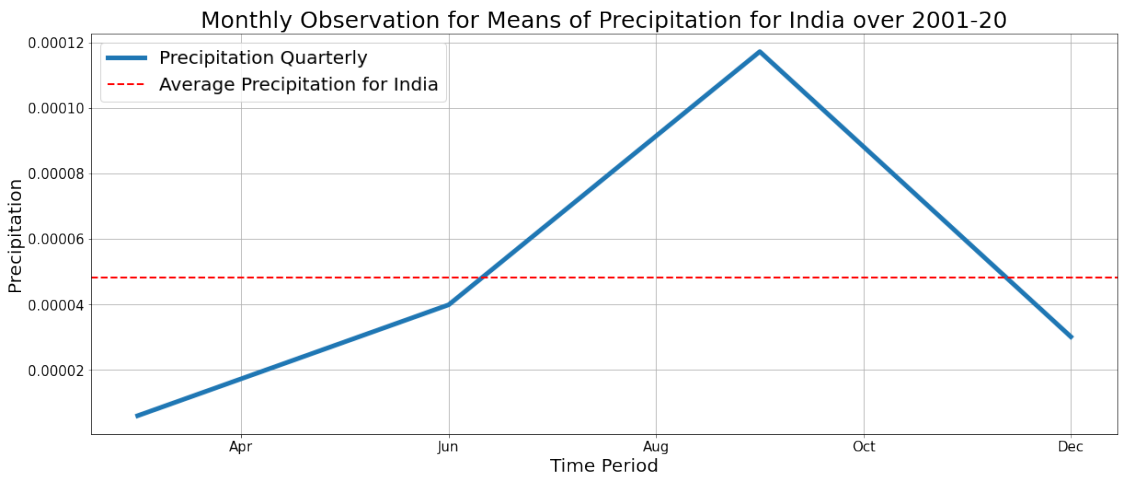
[8]:
```
precip_india = pd.read_csv("India_Final/precip.csv")
precip_india.State = precip_india.State.str.lower()
precip_india.date = pd.to_datetime(precip_india.date)
precip_india
```

[8]:

	State	date	precipitation
0	andaman & nicobar island	2001-03-31	0.000027
1	andaman & nicobar island	2001-06-30	0.000068
2	andaman & nicobar island	2001-09-30	0.000141
3	andaman & nicobar island	2001-12-31	0.000078
4	andaman & nicobar island	2002-03-31	NaN
…	…	…	…
2655	west bengal	2018-12-31	NaN
2656	west bengal	2019-03-31	0.000002
2657	west bengal	2019-06-30	0.000046
2658	west bengal	2019-09-30	0.000136
2659	west bengal	2019-12-31	0.000032

[2660 rows x 3 columns]

[9]:
```
# Visualizing by grouping into quarterly data
precip_india_date = precip_india.groupby("date").mean()
plt.figure(figsize = (20,8))
precip_india_date["month"] = precip_india_date.index.month
precip_india_date = precip_india_date.groupby("month").mean()
plt.plot( precip_india_date.index.tolist(),precip_india_date["precipitation"],␣
 ↪linewidth = 5, label = "Precipitation Quarterly")
plt.gca().axhline(y=precip_india_date["precipitation"].mean(),color='r',␣
 ↪linestyle='--', lw=2, label = "Average Precipitation for India")
plt.title("Monthly Observation for Means of Precipitation for India over␣
 ↪2001-20",fontsize = 25)
plt.xlabel("Time Period",fontsize= 20)
plt.ylabel("Precipitation",fontsize= 20)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
xlabels = ["Feb", "Apr", "Jun", "Aug", "Oct", "Dec"]
plt.gca().set_xticklabels(xlabels,fontsize = 15)
plt.legend(fontsize = 20)
plt.grid()
```



As we can observe the months of June - October receive plenty of Rainfall above the average. And there is retreating Monsoon for months of October and that is the reason, the precipitation is high

2.2 Merge and Visualisation For India

2.2.1 Merging into a bigger Dataframe

[10]:
```
mega= aod_india.merge(evaptrans_india,how = "inner", left_on =["ST_NM","date"],␣
 ↪right_on = ["ST_NM","date"])
# mega.set_index(["ST_NM", "date"])
mega = mega.merge(fire_india, left_on =["ST_NM","date"], right_on =␣
 ↪["State","date"])
mega= mega[["date","aod","evaptrans","fire","ST_NM"]]
mega
```

[10]:

	date	aod	evaptrans	fire	ST_NM
0	2001-01-01	192.204372	361.736801	4.525918	andaman & nicobar island
1	2001-04-01	244.904924	304.005540	4.408742	andaman & nicobar island
2	2001-07-01	343.086393	235.343897	4.313915	andaman & nicobar island
3	2001-10-01	243.661127	272.322672	4.460896	andaman & nicobar island
4	2002-01-01	250.867679	350.833868	4.568691	andaman & nicobar island
…	…	…	…	…	…
2609	2018-10-01	655.586612	123.843370	4.872379	west bengal
2610	2019-01-01	856.090348	83.715550	4.875533	west bengal
2611	2019-04-01	751.726200	107.010649	4.796782	west bengal
2612	2019-07-01	837.340381	222.791903	4.595394	west bengal
2613	2019-10-01	758.595713	176.575370	4.841796	west bengal

[2614 rows x 5 columns]

2.2.2 Summary Statistics

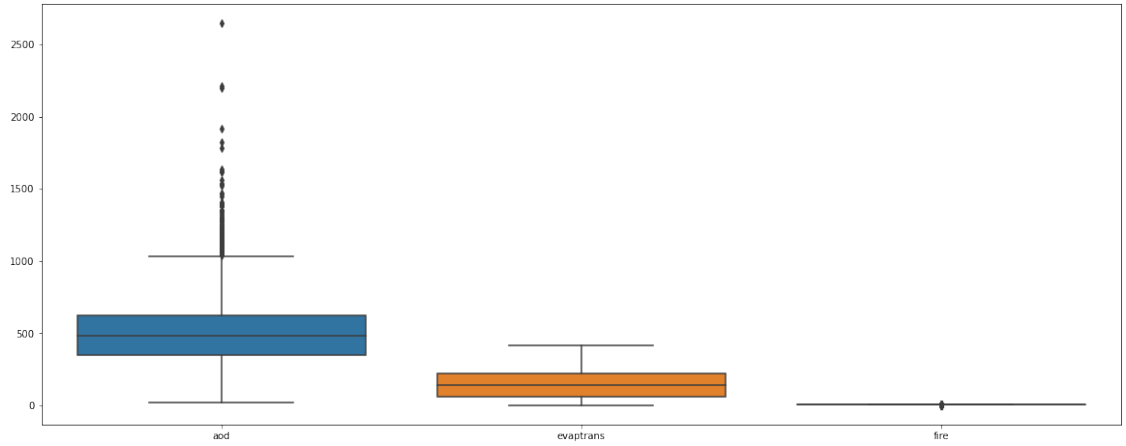
[11]:
```
mega.describe()
```

[11]:

	aod	evaptrans	fire
count	2614.000000	2614.000000	2614.000000
mean	513.521521	147.370414	4.717915
std	248.514758	95.080522	0.326085
min	20.000000	0.401821	3.011499
25%	350.545533	63.747778	4.631523
50%	482.070149	138.491256	4.812031
75%	624.470371	224.266556	4.910601
max	2649.076930	414.301047	4.996642

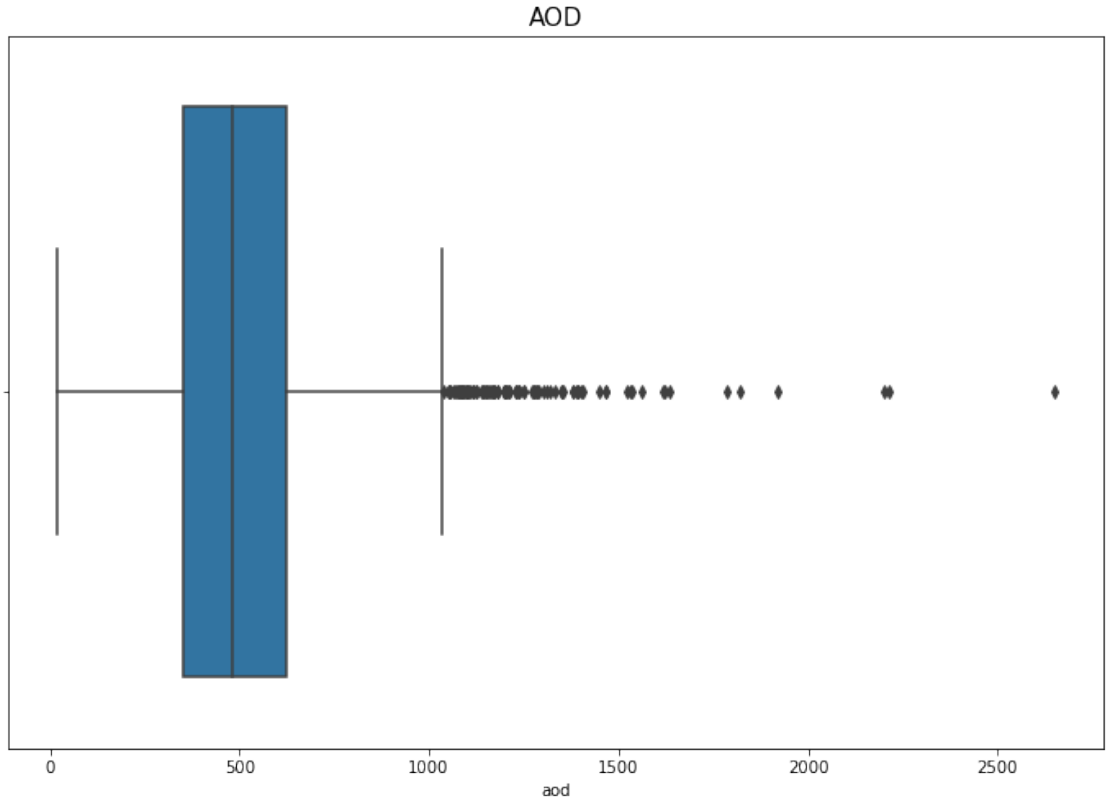
[12]:
```
plt.figure(figsize = (20,8))
sns.boxplot(data = mega)
```

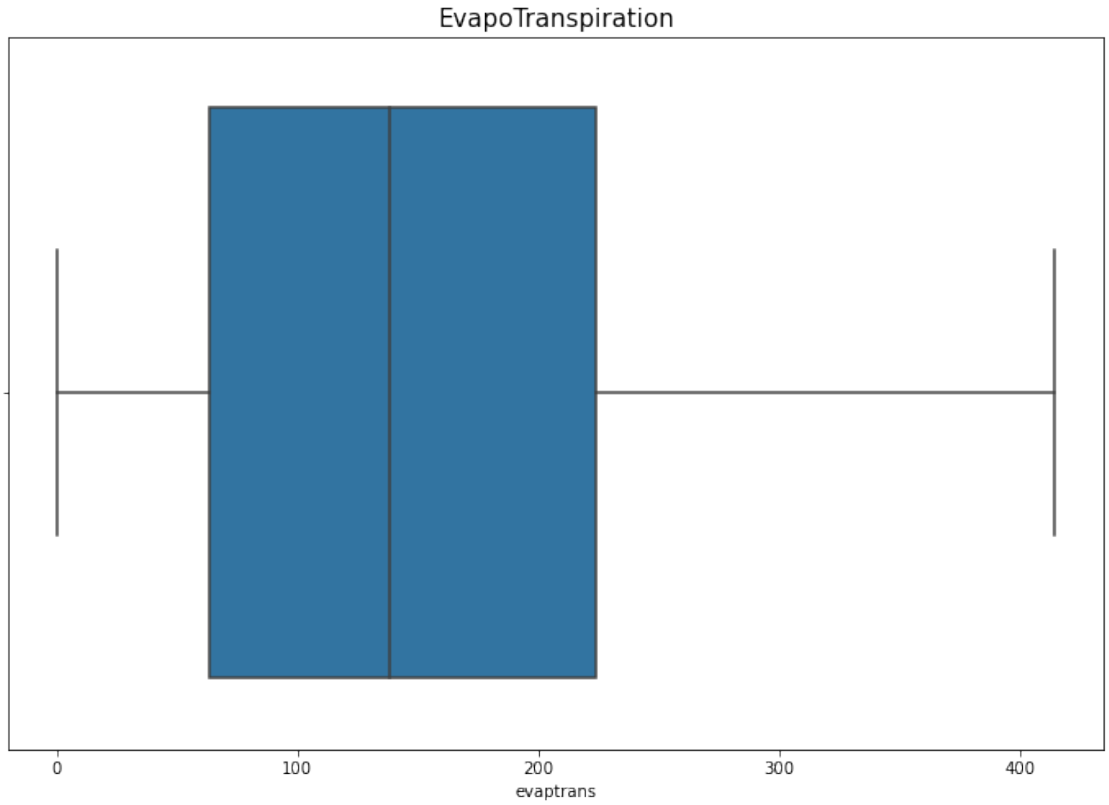
[12]: <AxesSubplot:>

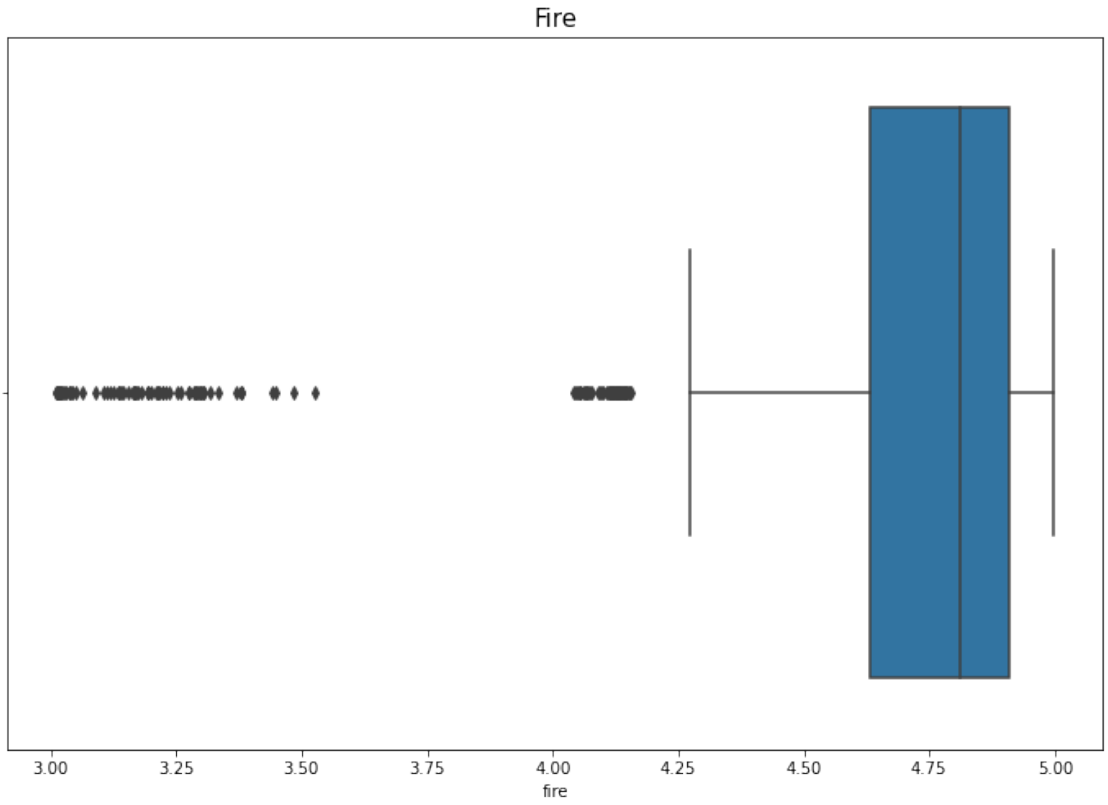


[13]:
```
plt.figure(figsize = (12,8))
plt.title("AOD", fontsize = 15)
sns.boxplot(mega["aod"])
plt.show()

plt.figure(figsize = (12,8))
plt.title("EvapoTranspiration", fontsize = 15)
sns.boxplot(mega["evaptrans"])
plt.show()

plt.figure(figsize = (12,8))
plt.title("Fire", fontsize = 15)
sns.boxplot(mega["fire"])
```

```
plt.show()
```




EvapoTranspiration
0
100
200
300
400
evaptrans



We can see that AOD values peak a lot and there are a lot of outliers.

There are no outliers in Evapotranspiration

There is very little variation in Fire.

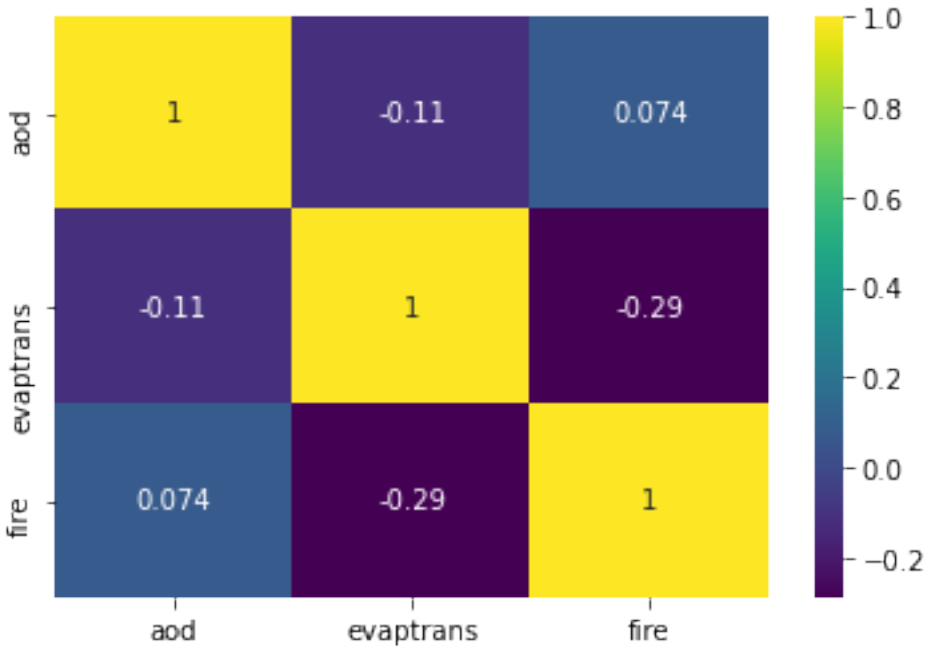
2.2.3 Correlation Matrix

[14]:
```
mega.corr()
```

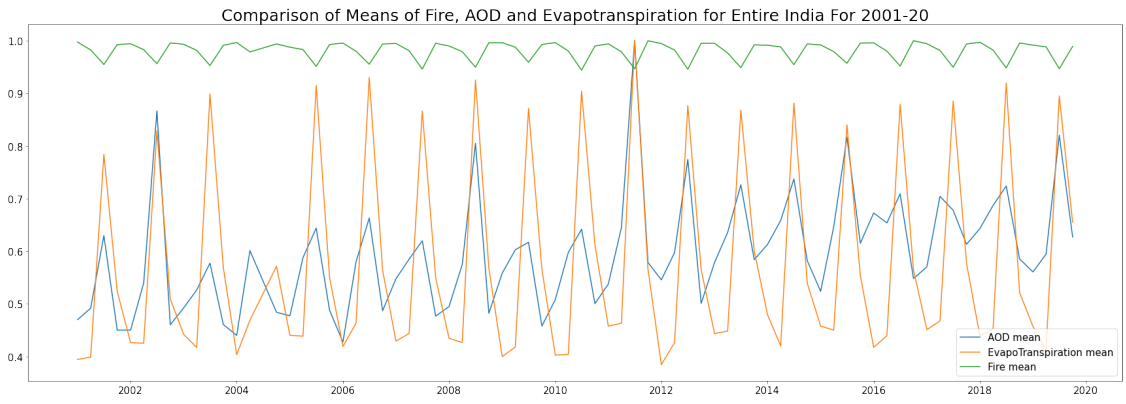
[14]:
```
                aod  evaptrans      fire
aod        1.000000  -0.110874  0.073843
evaptrans -0.110874   1.000000 -0.289024
fire       0.073843  -0.289024  1.000000
```

[15]:
```
sns.heatmap(mega.corr(),annot = True,  cmap = "viridis")
```

[15]:
```
<AxesSubplot:>
```



[16]:
```
# fig, axs = plt.subplots(nrows=3, figsize = (20,10))
plt.figure(figsize = (30,10))
india_mega = mega.groupby("date").mean()
plt.plot(india_mega.index.tolist(),india_mega.aod/max(india_mega.aod), label =␣
 ↪"AOD mean")
plt.plot(india_mega.index.tolist(),india_mega.evaptrans/max(india_mega.
 ↪evaptrans),label = "EvapoTranspiration mean")
plt.plot(india_mega.index.tolist(),india_mega.fire/max(india_mega.fire),label =␣
 ↪"Fire mean")
plt.legend(fontsize = 15,loc = 4)
plt.title("Comparison of Means of Fire, AOD and Evapotranspiration for Entire␣
 ↪India For 2001-20",fontsize = 25)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()
# axs[0].plot(india_mega.index.tolist(),india_mega.aod mean, label = "aod mean")
# axs[0].set_title("AOD")
# axs[1].plot(india_mega.index.tolist(),india_mega.evaptrans mean,label =␣
 ↪"evaptrans mean")
# axs[1].set_title("evaptrans mean")
# axs[2].plot(india_mega.index.tolist(),india_mega.fire mean,label = "fire␣
 ↪mean")
# axs[2].set_title("fire mean")
```



This is the correlation matrix for the above variables. India being a large contry and covering various climatic regions, shows a large variation in the data among its districts. as a conclusive corelation could not be attained from the data.

2.2.4 Plotting Chloropleths for Individual Variables

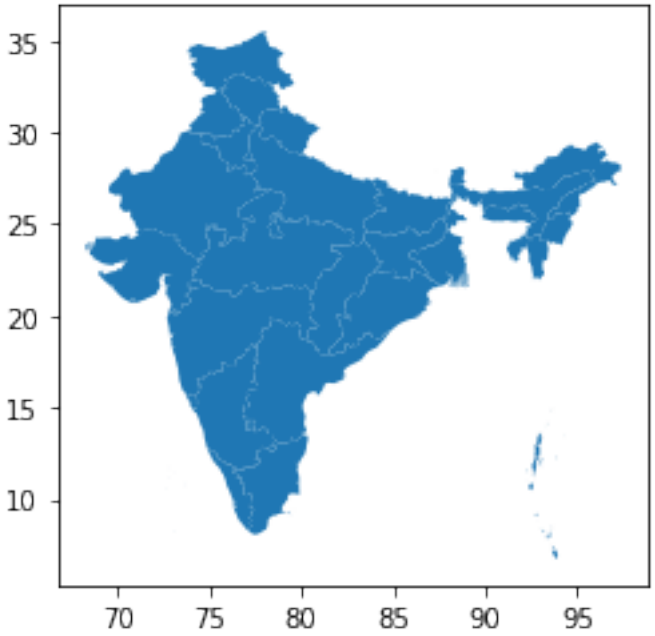
[17]:
```
# Loading India State Shapefile
India_shape = gpd.read_file("india_state.shp")
India_shape["NAME_1"] = India_shape["NAME_1"].str.lower()
India_shape.plot()
# print(len(India_shape.NAME_1.unique()))
India_shape = India_shape[["NAME_1","geometry"]]
states = ['andaman & nicobar island', 'andhra pradesh', 'arunanchal pradesh',
       'assam', 'bihar', 'chandigarh', 'chhattisgarh',
       'dadara & nagar havelli', 'daman & diu','nct of delhi' ,'goa', 'gujarat',
       'haryana', 'himachal pradesh', 'jammu & kashmir', 'jharkhand',
       'karnataka', 'kerala', 'lakshadweep', 'madhya pradesh',
       'maharashtra', 'manipur', 'meghalaya', 'mizoram', 'nagaland',
     'odisha', 'puducherry', 'punjab', 'rajasthan',
       'sikkim', 'tamil nadu', 'tripura', 'uttar pradesh', 'uttarakhand',
       'west bengal']
# print(len(states))
# print(states)
India_shape["states"] = states
India_shape
```

[17]:
```
                   NAME_1                                           geometry  \
0      andaman and nicobar  MULTIPOLYGON (((93.78773 6.85264, 93.78849 6.8…
1           andhra pradesh  MULTIPOLYGON (((80.27458 13.45958, 80.27458 13…
2        arunachal pradesh  POLYGON ((95.23763 26.68629, 95.23598 26.68823…
3                    assam  MULTIPOLYGON (((89.87145 25.53730, 89.87118 25…
4                    bihar  MULTIPOLYGON (((88.10622 26.53601, 88.10202 26…
5               chandigarh  POLYGON ((76.81051 30.68495, 76.80592 30.68026…
```

```
6             chhattisgarh  POLYGON ((84.00250 22.52086, 83.99429 22.52881…
7   dadra and nagar haveli  POLYGON ((72.99046 20.29209, 73.00357 20.29314…
8            daman and diu  MULTIPOLYGON (((72.86590 20.46751, 72.86238 20…
9                    delhi  POLYGON ((77.19444 28.80229, 77.19692 28.79528…
10                     goa  MULTIPOLYGON (((73.78181 15.35569, 73.78181 15…
11                 gujarat  MULTIPOLYGON (((70.87041 20.71872, 70.86694 20…
12                 haryana  POLYGON ((77.57489 30.38452, 77.58444 30.38136…
13       himachal pradesh   POLYGON ((78.41076 32.51174, 78.41498 32.51100…
14      jammu and kashmir   POLYGON ((78.41076 32.51174, 78.40917 32.51202…
15               jharkhand  POLYGON ((87.78758 25.22049, 87.78519 25.20908…
16               karnataka  MULTIPOLYGON (((74.67097 13.19986, 74.67097 13…
17                  kerala  MULTIPOLYGON (((76.46736 9.54097, 76.46736 9.5…
18             lakshadweep  MULTIPOLYGON (((73.01014 8.28042, 73.01014 8.2…
19          madhya pradesh  POLYGON ((82.80778 23.96306, 82.79865 23.96218…
20             maharashtra  MULTIPOLYGON (((73.45597 15.88986, 73.45597 15…
21                 manipur  POLYGON ((94.68243 25.45973, 94.68230 25.45659…
22               meghalaya  POLYGON ((92.42590 25.03182, 92.42393 25.03284…
23                 mizoram  POLYGON ((93.00050 24.40327, 93.02435 24.39047…
24                nagaland  POLYGON ((95.23763 26.68629, 95.23450 26.68307…
25                  orissa  MULTIPOLYGON (((84.76986 19.10597, 84.76986 19…
26              puducherry  MULTIPOLYGON (((79.84486 10.82653, 79.84486 10…
27                  punjab  POLYGON ((76.76986 30.90639, 76.76254 30.89341…
28               rajasthan  POLYGON ((74.51907 29.94335, 74.51279 29.93004…
29                  sikkim  POLYGON ((88.75169 27.14821, 88.73718 27.13952…
30              tamil nadu  MULTIPOLYGON (((77.55596 8.07903, 77.55596 8.0…
31                 tripura  POLYGON ((92.23100 24.50409, 92.23427 24.50044…
32           uttar pradesh  POLYGON ((79.76197 28.88969, 79.77447 28.89183…
33             uttaranchal  POLYGON ((80.07238 28.82974, 80.07025 28.82964…
34             west bengal  MULTIPOLYGON (((88.01861 21.57278, 88.01889 21…
```

```
                      states
0   andaman & nicobar island
1             andhra pradesh
2          arunanchal pradesh
3                       assam
4                       bihar
5                  chandigarh
6                chhattisgarh
7     dadara & nagar havelli
8                 daman & diu
9                nct of delhi
10                        goa
11                    gujarat
12                    haryana
13           himachal pradesh
14            jammu & kashmir
15                  jharkhand
```

```
16              karnataka
17                 kerala
18            lakshadweep
19         madhya pradesh
20            maharashtra
21                manipur
22              meghalaya
23                mizoram
24               nagaland
25                 odisha
26             puducherry
27                 punjab
28              rajasthan
29                 sikkim
30             tamil nadu
31                tripura
32          uttar pradesh
33            uttarakhand
34            west bengal
```



[18]:
```
# Evapotranspiration
evaptrans_india_g = evaptrans_india.groupby(by="ST_NM").mean()
evaptrans_india_g
```

[18]:

ST_NM	evaptrans
andaman & nicobar island	304.190187
andhra pradesh	125.869246
arunanchal pradesh	192.820680
assam	180.686646
bihar	104.976782
chandigarh	64.423658
chhattisgarh	134.087032
dadara & nagar havelli	123.050228
daman & diu	113.625327
goa	242.137213
gujarat	87.775922
haryana	69.414448
himachal pradesh	110.329185
jammu & kashmir	98.325873
jharkhand	113.620325
karnataka	142.956591
kerala	292.192933
lakshadweep	159.216864
madhya pradesh	91.315351
maharashtra	110.205078
manipur	227.655612
meghalaya	236.946064
mizoram	283.092522
nagaland	242.448732
nct of delhi	40.431817
odisha	143.733817
puducherry	195.357085
punjab	74.554802
rajasthan	49.651434
sikkim	155.911404
tamil nadu	142.113788
tripura	251.036554
uttar pradesh	78.702927
uttarakhand	115.326198
west bengal	133.358110

[19]:
```
# Merging the dataframes
merged_n = mega.groupby(["ST_NM"]).mean()
# Precipitation Variable
precip_india_g = precip_india.groupby(["State"]).mean()
precip_india_g["States"] = states
# print(precip_india_g)
merged_k = precip_india_g.merge(merged_n,left_on=["States"],right_on=["ST_NM"])
merged_t = India_shape.merge(merged_k,left_on=["states"],right_on=["States"])
merged_t
```

[19]:

	NAME_1	geometry
0	andaman and nicobar	MULTIPOLYGON (((93.78773 6.85264, 93.78849 6.8…
1	andhra pradesh	MULTIPOLYGON (((80.27458 13.45958, 80.27458 13…
2	arunachal pradesh	POLYGON ((95.23763 26.68629, 95.23598 26.68823…
3	assam	MULTIPOLYGON (((89.87145 25.53730, 89.87118 25…
4	bihar	MULTIPOLYGON (((88.10622 26.53601, 88.10202 26…
5	chandigarh	POLYGON ((76.81051 30.68495, 76.80592 30.68026…
6	chhattisgarh	POLYGON ((84.00250 22.52086, 83.99429 22.52881…
7	dadra and nagar haveli	POLYGON ((72.99046 20.29209, 73.00357 20.29314…
8	daman and diu	MULTIPOLYGON (((72.86590 20.46751, 72.86238 20…
9	delhi	POLYGON ((77.19444 28.80229, 77.19692 28.79528…
10	goa	MULTIPOLYGON (((73.78181 15.35569, 73.78181 15…
11	gujarat	MULTIPOLYGON (((70.87041 20.71872, 70.86694 20…
12	haryana	POLYGON ((77.57489 30.38452, 77.58444 30.38136…
13	himachal pradesh	POLYGON ((78.41076 32.51174, 78.41498 32.51100…
14	jammu and kashmir	POLYGON ((78.41076 32.51174, 78.40917 32.51202…
15	jharkhand	POLYGON ((87.78758 25.22049, 87.78519 25.20908…
16	karnataka	MULTIPOLYGON (((74.67097 13.19986, 74.67097 13…
17	kerala	MULTIPOLYGON (((76.46736 9.54097, 76.46736 9.5…
18	lakshadweep	MULTIPOLYGON (((73.01014 8.28042, 73.01014 8.2…
19	madhya pradesh	POLYGON ((82.80778 23.96306, 82.79865 23.96218…
20	maharashtra	MULTIPOLYGON (((73.45597 15.88986, 73.45597 15…
21	manipur	POLYGON ((94.68243 25.45973, 94.68230 25.45659…
22	meghalaya	POLYGON ((92.42590 25.03182, 92.42393 25.03284…
23	mizoram	POLYGON ((93.00050 24.40327, 93.02435 24.39047…
24	nagaland	POLYGON ((95.23763 26.68629, 95.23450 26.68307…
25	orissa	MULTIPOLYGON (((84.76986 19.10597, 84.76986 19…
26	puducherry	MULTIPOLYGON (((79.84486 10.82653, 79.84486 10…
27	punjab	POLYGON ((76.76986 30.90639, 76.76254 30.89341…
28	rajasthan	POLYGON ((74.51907 29.94335, 74.51279 29.93004…
29	sikkim	POLYGON ((88.75169 27.14821, 88.73718 27.13952…
30	tamil nadu	MULTIPOLYGON (((77.55596 8.07903, 77.55596 8.0…
31	tripura	POLYGON ((92.23100 24.50409, 92.23427 24.50044…
32	uttar pradesh	POLYGON ((79.76197 28.88969, 79.77447 28.89183…
33	uttaranchal	POLYGON ((80.07238 28.82974, 80.07025 28.82964…
34	west bengal	MULTIPOLYGON (((88.01861 21.57278, 88.01889 21…

	states	precipitation	States
0	andaman & nicobar island	0.000085	andaman & nicobar island
1	andhra pradesh	0.000029	andhra pradesh
2	arunanchal pradesh	0.000052	arunanchal pradesh
3	assam	0.000085	assam
4	bihar	0.000041	bihar
5	chandigarh	0.000027	chandigarh
6	chhattisgarh	0.000039	chhattisgarh
7	dadara & nagar havelli	0.000061	dadara & nagar havelli
8	daman & diu	0.000042	daman & diu

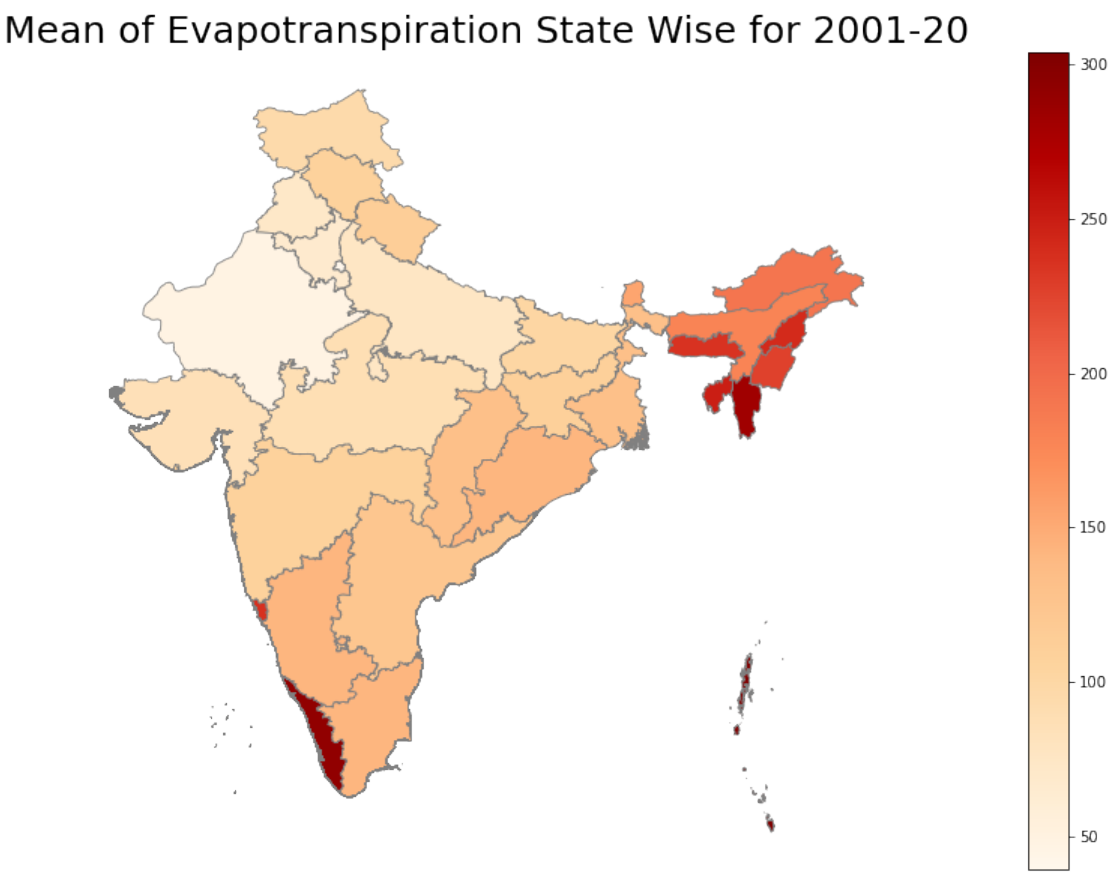
9	nct of delhi	0.000023	nct of delhi
10	goa	0.000069	goa
11	gujarat	0.000024	gujarat
12	haryana	0.000019	haryana
13	himachal pradesh	0.000026	himachal pradesh
14	jammu & kashmir	0.000018	jammu & kashmir
15	jharkhand	0.000036	jharkhand
16	karnataka	0.000033	karnataka
17	kerala	0.000069	kerala
18	lakshadweep	0.000031	lakshadweep
19	madhya pradesh	0.000031	madhya pradesh
20	maharashtra	0.000035	maharashtra
21	manipur	0.000077	manipur
22	meghalaya	0.000136	meghalaya
23	mizoram	0.000096	mizoram
24	nagaland	0.000069	nagaland
25	odisha	0.000041	odisha
26	puducherry	0.000053	puducherry
27	punjab	0.000019	punjab
28	rajasthan	0.000017	rajasthan
29	sikkim	0.000062	sikkim
30	tamil nadu	0.000033	tamil nadu
31	tripura	0.000099	tripura
32	uttar pradesh	0.000030	uttar pradesh
33	uttarakhand	0.000028	uttarakhand
34	west bengal	0.000055	west bengal

	aod	evaptrans	fire
0	306.982954	303.933168	4.434463
1	562.144194	124.540109	4.775461
2	214.301950	191.366408	4.665618
3	447.609676	179.312599	4.719662
4	715.951401	101.710861	4.836517
5	599.023342	62.724581	4.900748
6	548.563571	132.083348	4.815132
7	592.162697	102.300058	4.820137
8	562.312829	108.600739	4.112544
9	854.701378	39.272583	4.906230
10	614.207351	237.444728	4.714590
11	615.556306	85.606798	4.849315
12	766.034309	67.897917	4.939030
13	273.371955	108.095492	4.752303
14	252.277817	95.924405	4.712951
15	697.668065	110.989286	4.836209
16	507.579702	142.287281	4.779887
17	506.184227	292.538598	4.658871
18	413.743696	159.538613	3.189081

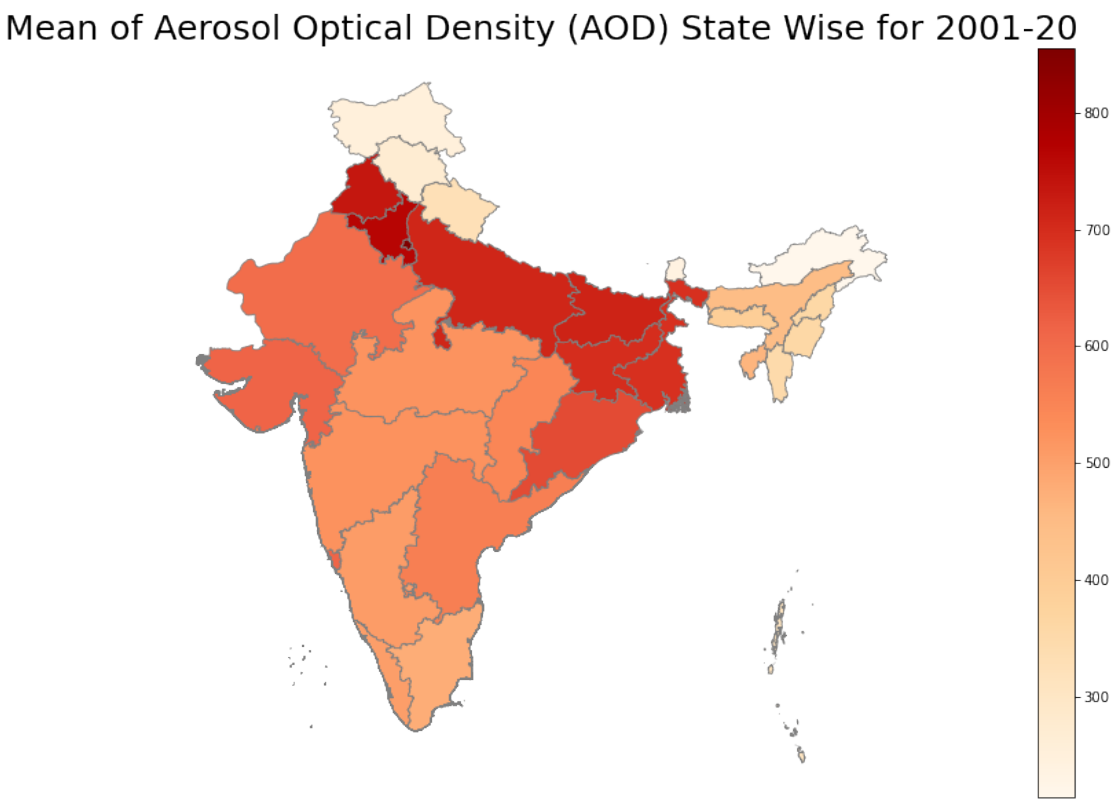
```
19  522.361304   88.419785  4.858980
20  524.892543  108.107783  4.771212
21  361.602054  226.875316  4.771881
22  395.724237  236.416509  4.828653
23  351.227393  282.285918  4.826877
24  359.800802  241.687594  4.819603
25  654.244848  141.800187  4.745042
26  512.111937  195.200282  4.518500
27  735.359309   72.850962  4.934328
28  598.632359   48.150555  4.916460
29  240.504411  154.716485  4.673888
30  479.440503  141.937417  4.759540
31  465.388682  250.088878  4.819282
32  712.010758   76.085011  4.880778
33  328.070316  112.821891  4.804643
34  693.228271  131.429884  4.781339
```

[20]:
```
# Plotting For EvapoTranspiration
variable = 'evaptrans'
vmin, vmax = merged_t.evaptrans.min(), merged_t.evaptrans.max()
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
ax.set_title('Mean of Evapotranspiration State Wise for 2001-20',␣
 ↪fontdict={'fontsize': '25', 'fontweight' : '3'})
sm = plt.cm.ScalarMappable(cmap='OrRd', norm=plt.Normalize(vmin=vmin,␣
 ↪vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
merged_t.plot(column=variable, cmap='OrRd', linewidth=0.8, ax=ax, edgecolor='0.
 ↪5')
```

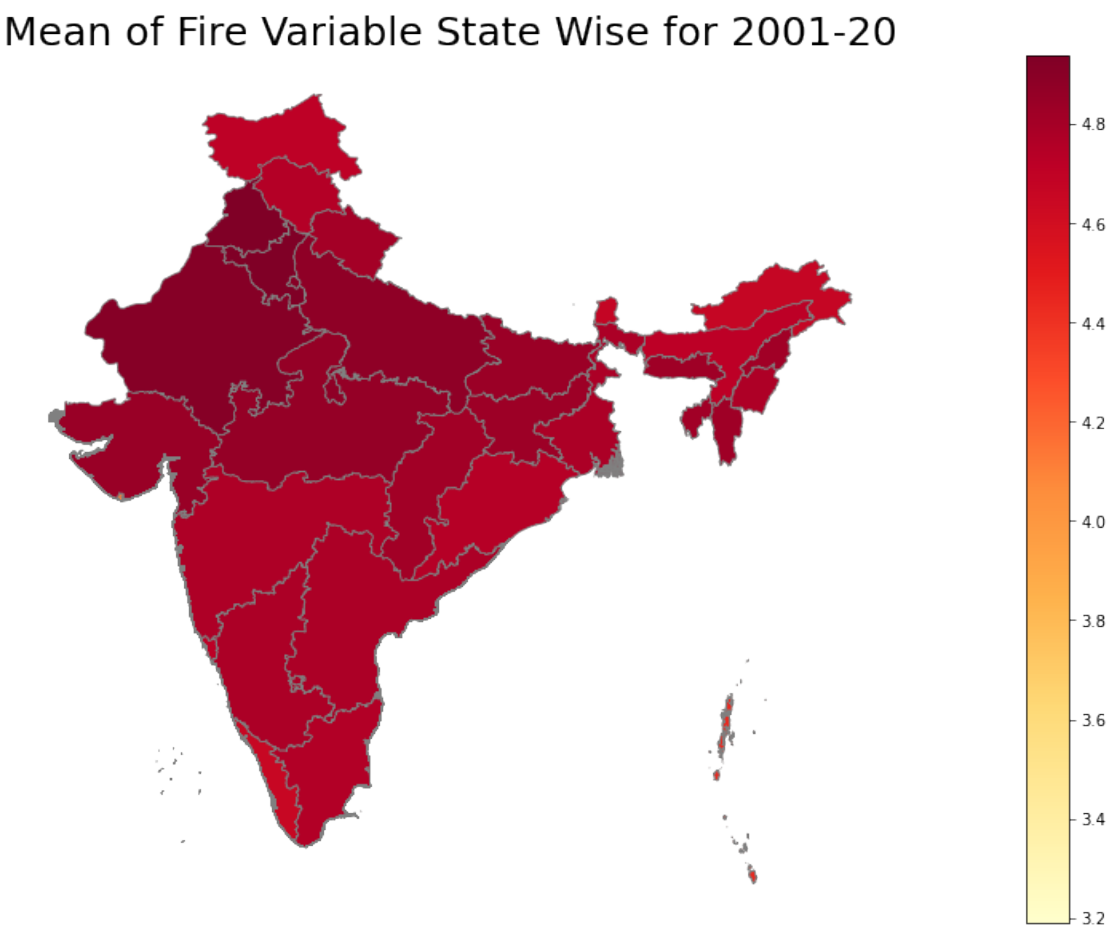
[20]:
```
<AxesSubplot:title={'center':'Mean of Evapotranspiration State Wise for
2001-20'}>
```



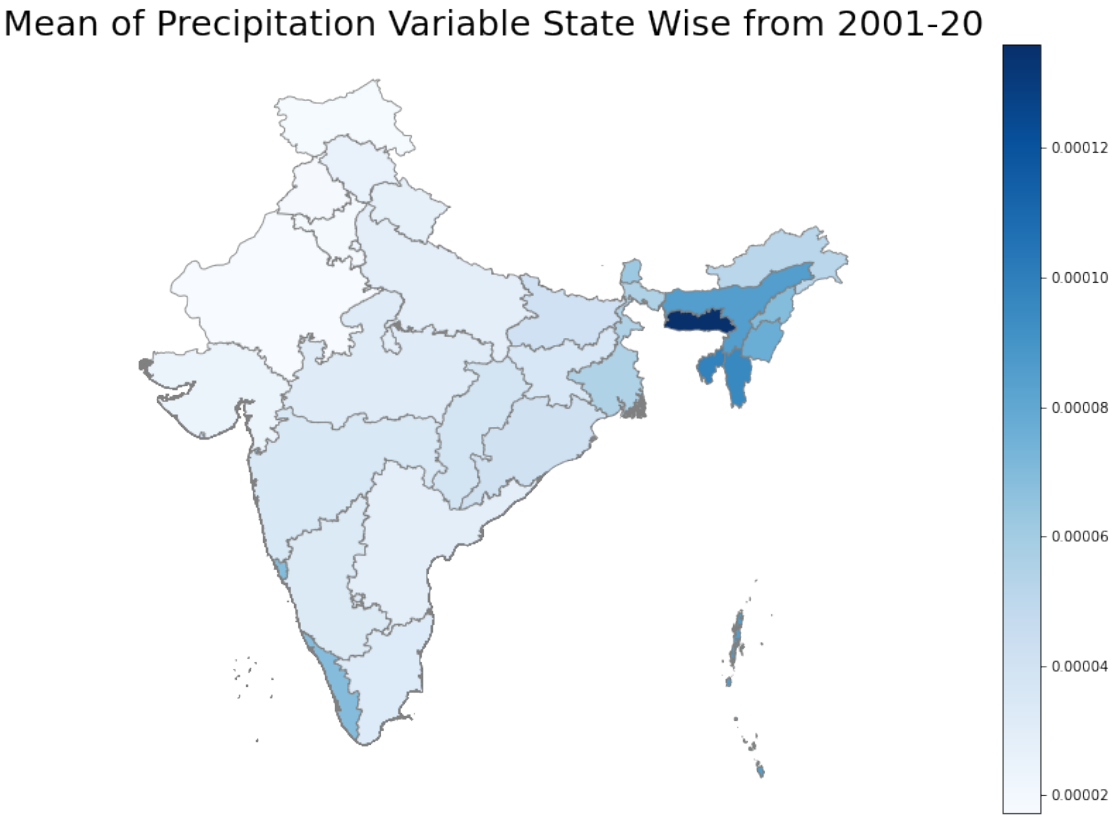
[21]:
```
# For AOD
variable = 'aod'
vmin, vmax = merged_t.aod.min(), merged_t.aod.max()
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
ax.set_title('Mean of Aerosol Optical Density (AOD) State Wise for 2001-20',␣
 ↪fontdict={'fontsize': '25', 'fontweight' : '3'})
sm = plt.cm.ScalarMappable(cmap='OrRd', norm=plt.Normalize(vmin=vmin,␣
 ↪vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
merged_t.plot(column=variable, cmap='OrRd', linewidth=0.8, ax=ax, edgecolor='0.
 ↪5')
plt.show()
```



[22]:
```
# For Fire
variable = 'fire'
vmin, vmax = merged_t.fire.min(), merged_t.fire.max()
# print(vmin,vmax)
# print(merged_t.fire_.min(), merged_t.fire_india.max())
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
ax.set_title('Mean of Fire Variable State Wise for 2001-20',␣
 ↪fontdict={'fontsize': '25', 'fontweight' : '3'})
sm = plt.cm.ScalarMappable(cmap='YlOrRd', norm=plt.Normalize(vmin=vmin,␣
 ↪vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
merged_t.plot(column=variable, cmap='YlOrRd',vmin = vmin, vmax = vmax ,␣
 ↪linewidth=0.8, ax=ax, edgecolor='0.5')
plt.show()
```



[23]:
```
# Precipitation Variable
variable = 'precipitation'
vmin, vmax = merged_t.precipitation.min(), merged_t.precipitation.max()
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
ax.set_title('Mean of Precipitation Variable State Wise from 2001-20',␣
↪fontdict={'fontsize': '25', 'fontweight' : '3'})
sm = plt.cm.ScalarMappable(cmap='Blues', norm=plt.Normalize(vmin=vmin,␣
↪vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
merged_t.plot(column=variable, cmap='Blues', linewidth=0.8, ax=ax, edgecolor='0.
↪5')
plt.show()
```



3 Specific City Case Study - Bangalore

As a good corelation could not be obtained from the data of the Indias districts, we dive into the analysis of the variables in a smaller region. The region chosen for this analysis is Bangalore

3.1 Loading Data

3.1.1 Loading Wards List (Ward Numbers and Names)

[24]:
```
Bangalore_Wards_List = pd.read_excel('Bangalore/Bangalore_Wards_List.xlsx')
Bangalore_Wards_List.set_index("Ward_No",inplace = True)
Bangalore_Wards_List
```

[24]:

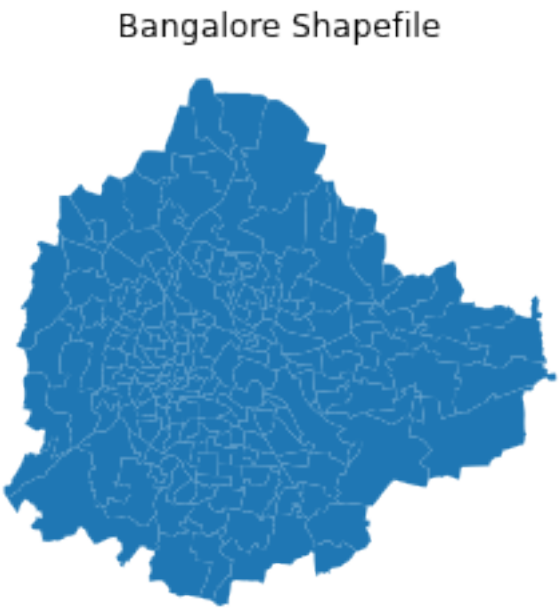
Ward_No	Ward_Name
1	Kempegowda Ward
2	Chowdeswari Ward
3	Atturu
4	Yelahanka Satellite Town
5	Jakkuru
…	…

```
194                    Gottigere
195                   Konankunte
196                   Anjanapura
197                  Vasanthpura
198                  Hemmigepura

[198 rows x 1 columns]
```

3.1.2 Loading Bangalore Shapefile

[25]:
```
# loading the shapefile
path = r'Cleaned CSVS/Bangalore/BBMP-polygon.shp'
map_Bangalore = gpd.read_file(path)
# Viewing the shapefile
map_Bangalore.plot()
plt.axis("off")
plt.title("Bangalore Shapefile")
```

[25]:
```
Text(0.5, 1.0, 'Bangalore Shapefile')
```



3.1.3 Loading Datasets

Variable 1: Aerosol Optical Density (AOD) - MODIS AOD

Aerosol optical depth is a measure of the extinction of the solar beam by dust and haze. In other words, particles in the atmosphere (dust, smoke, pollution) can block sunlight by absorbing or by

scattering light. AOD tells us how much direct sunlight is prevented from reaching the ground by these aerosol particles.

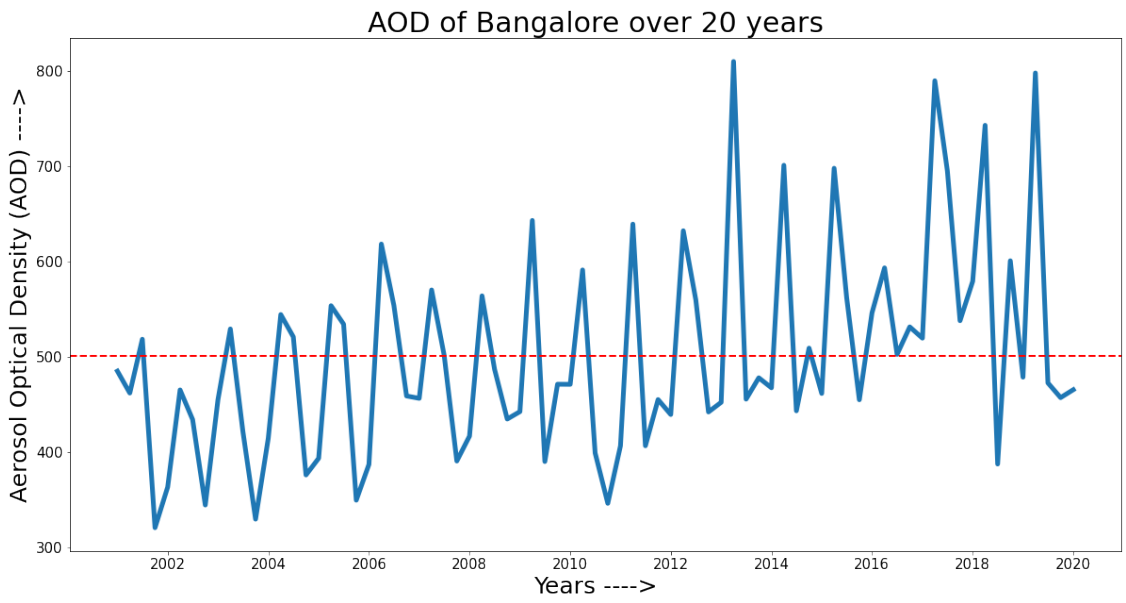
[26]:
```
AOD = pd.read_csv("Bangalore/AOD_2001-20.csv")
AOD.date = pd.to_datetime(AOD.date)
AOD.rename(columns={'mean':'AOD'}, inplace=True)
AOD
```

[26]:

	WARD_NO	date	AOD
0	2	2001-01-01	490.087543
1	3	2001-01-01	509.689948
2	4	2001-01-01	469.909609
3	51	2001-01-01	528.509737
4	53	2001-01-01	541.013335
…	…	…	…
14999	151	2020-01-01	432.935301
15000	152	2020-01-01	428.172094
15001	178	2020-01-01	453.492220
15002	173	2020-01-01	429.125293
15003	176	2020-01-01	441.046955

[15004 rows x 3 columns]

[27]:
```
# Univariate Plotting
AOD_date = AOD.groupby("date").mean()
plt.figure(figsize = (20,10))
plt.plot(AOD_date.index,AOD_date.AOD , lw = 5)
plt.title("AOD of Bangalore over 20 years", fontsize = 30)
plt.ylabel("Aerosol Optical Density (AOD) ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(AOD_date["AOD"].mean(), color = "r", lw = 2, linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()
```



The plot show the AOD values of the Bangalore region for the last 20 years. The graph shows that the AOD values have increased in the last 20 years, signifying increase in pollution in Bangalore.

[28]:
```
# Dividing Datasets for plotting
AOD_till_2010 = AOD[(AOD["date"]>= "20010101") & (AOD["date"]< "20110101")]
AOD_till_2010 = AOD_till_2010.groupby("WARD_NO").mean()
AOD_from_2011 = AOD[(AOD["date"]>= "20110101")]
AOD_from_2011 = AOD_from_2011.groupby("WARD_NO").mean()
AOD_wards = AOD.groupby("WARD_NO").mean()
```

[29]:
```
merged_Bangalore_till_2010_AOD = map_Bangalore.merge(AOD_till_2010, left_on =␣
↪"WARD_NO",right_index = True)
merged_Bangalore_from_2011_AOD = map_Bangalore.merge(AOD_from_2011, left_on =␣
↪"WARD_NO",right_index = True)
merged_Bangalore_AOD       = map_Bangalore.merge(AOD_wards, left_on =␣
↪"WARD_NO",right_index = True)
```

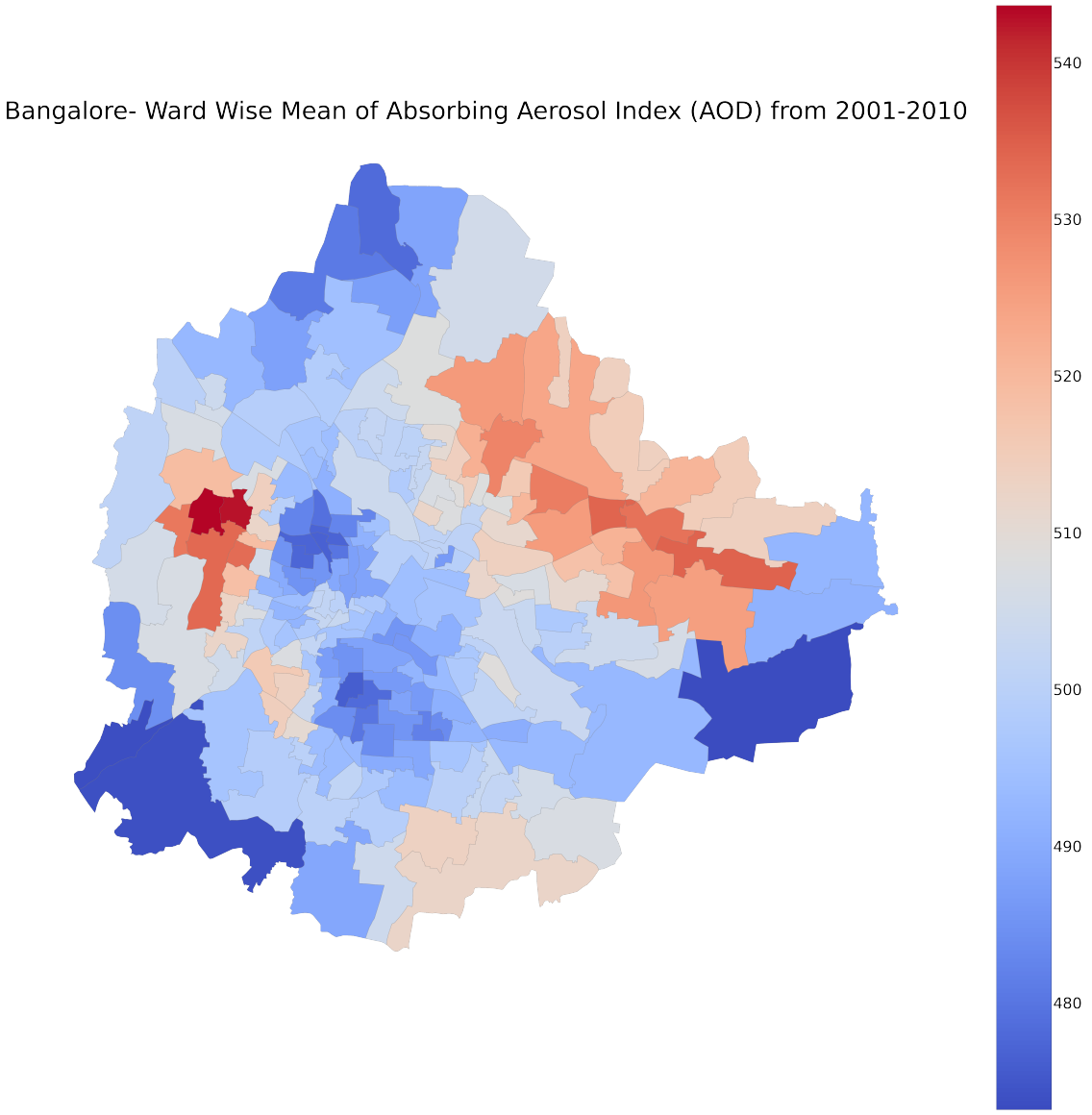
[30]:
```
# Storing the Wards which have observed a 25 % increase in AOD value in the␣
↪last decade
AOD_worse_wards = ((AOD_from_2011["AOD"] - AOD_till_2010["AOD"])/
↪AOD_till_2010["AOD"] ).tolist()
lst = []
lst1 = []
for i in range(len(AOD_worse_wards)):
    if(AOD_worse_wards[i] > 0.25):
        lst.append((Bangalore_Wards_List.loc[i][0]))
        lst1.append(i)
```

```
lst
```

[30]:
```
['Attiguppe',
 'Hampi Nagar',
 'Bapuji Nagar',
 'Jayanagar',
 'Srinagar',
 'Deepanjali Nagar',
 'Jakkasandra',
 'HSR Layout',
 'Bommanahalli',
 'Banashankari Temple ward',
 'Kumaraswamy Layout',
 'Chikkalsandra',
 'Uttarahalli',
 'Jaraganahalli',
 'Puttenahalli',
 'Bilekhalli',
 'Gottigere',
 'Anjanapura']
```

[31]:
```
# For 2001-10

variable = 'AOD'
vmin, vmax = merged_Bangalore_AOD["AOD"].min(), merged_Bangalore_AOD["AOD"].
 ↪max()
fig, ax = plt.subplots(1, figsize=(100, 100))
ax.axis('off')

ax.set_title('Bangalore- Ward Wise Mean of Absorbing Aerosol Index (AOD) from␣
 ↪2001-2010',
             fontdict={'fontsize': '120', 'fontweight' : '3'})


sm = plt.cm.ScalarMappable(cmap='coolwarm',norm=plt.Normalize(vmin=vmin
                                                              ,vmax=vmax ))
# empty array for the data range
sm.set_array([])

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
merged_Bangalore_till_2010_AOD.plot(column=variable, cmap='coolwarm',␣
 ↪linewidth=0.8, ax=ax,
                                    edgecolor='0.5', figsize = (150,450))
plt.show()
```

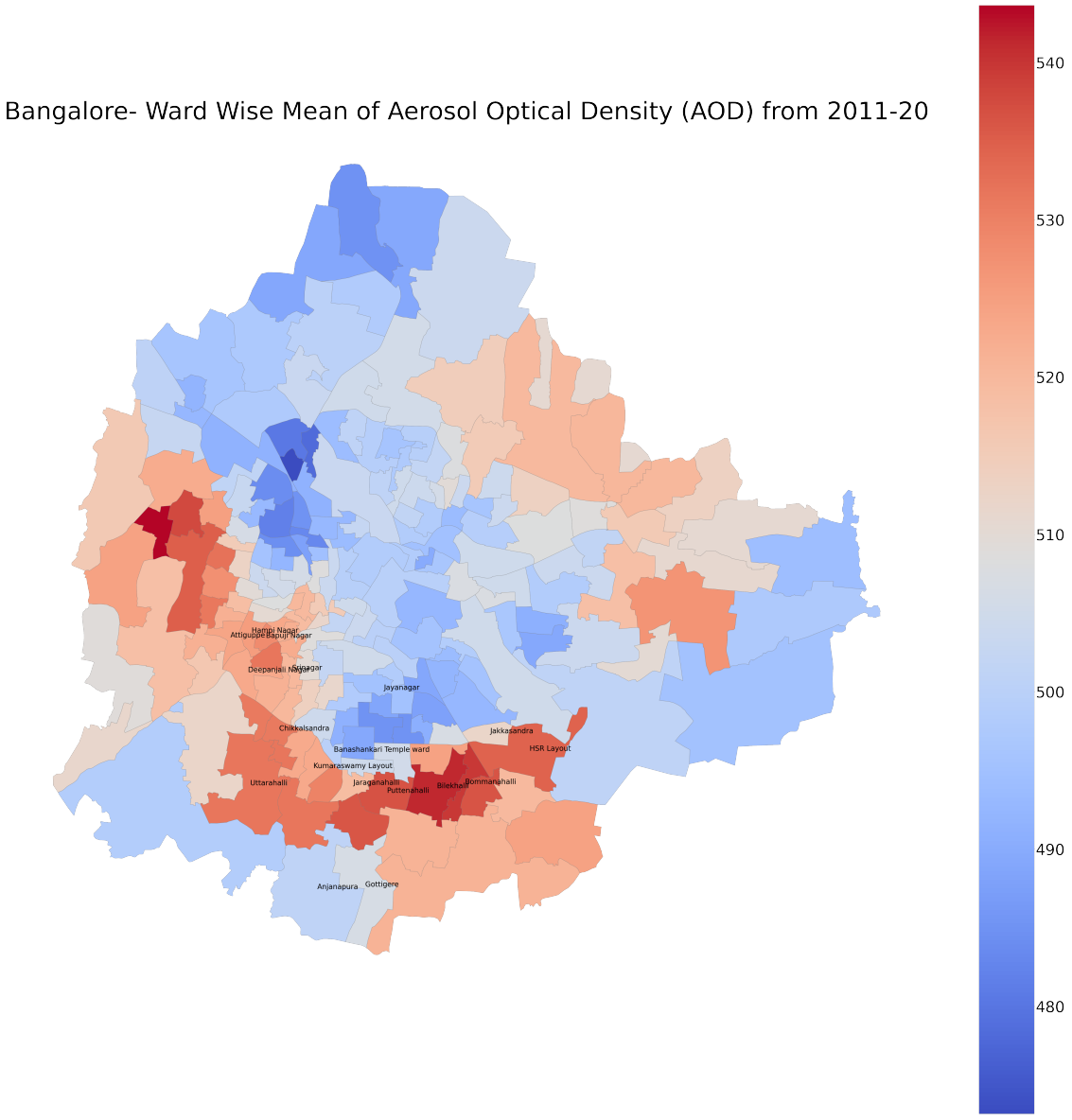
```
# fig, ax = plt.subplots(nrows=2,ncols=1,figsize=(75,50))
# # fig.suptitle('BANGALORE',fontsize=30,fontweight=5)

# variable = 'AOD'
# vmin, vmax = merged_Bangalore_AOD["AOD"].min(), merged_Bangalore_AOD["AOD"].
 ↪max()
# ax[0].axis('off')
# ax[0].set_title('Bangalore- Ward Wise Mean of Absorbing Aerosol Index (AOD)␣
 ↪from 2001-2010',
#              fontdict={'fontsize': '50', 'fontweight' : '3'})

# # ax[0].set_title('NDVI',fontdict={'fontsize': '25', 'fontweight' : '2'},y=-0.
 ↪05)
# # cmin = bang_wardmap['NDVI'].min()
# # cmax = bang_wardmap['NDVI'].max()
# cb = plt.cm.ScalarMappable(cmap='Reds', norm=plt.
 ↪Normalize(vmin=vmin,vmax=vmax))
# cb.set_array([])
# bx = fig.colorbar(cb,ax=ax[0])
# # bx.ax.tick_params(labelsize=75)
# merged_Bangalore_till_2010_AOD.plot(column=variable ,ax=ax[0],cmap='Reds',
#                                     edgecolor='0.5')

# ax[1].axis('off')
# ax[1].set_title('Bangalore- Ward Wise Mean of Absorbing Aerosol Index (AOD)␣
 ↪from 2011-2020',
#              fontdict={'fontsize': '50', 'fontweight' : '3'})
# cb = plt.cm.ScalarMappable(cmap='Reds', norm=plt.
 ↪Normalize(vmin=vmin,vmax=vmax))
# cb.set_array([])
# fig.colorbar(cb,ax=ax[1])
# dx = merged_Bangalore_from_2011_AOD.plot(column=variable␣
 ↪,ax=ax[1],cmap='Reds',
#                                     edgecolor='0.5')
# count = 0
# for x, y, label in zip(merged_Bangalore_from_2011_AOD.centroid.x,␣
 ↪merged_Bangalore_from_2011_AOD.centroid.y,
#                        merged_Bangalore_from_2011_AOD.WARD_NAME ):
#     if label not in lst:
#         continue
#     count = count + 1
# #     print(x,y,label)
#     dx.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 30)
# plt.show()
```



The chloropleth plotted here shows the spread of the AOD values in the wards of Bangalore for the time period of 2001-2010

[32]:
```
# For 2011-20
# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Aerosol Optical Density (AOD) from␣
↪2011-20'
             ,fontdict={'fontsize': '120', 'fontweight' : '3'})
```

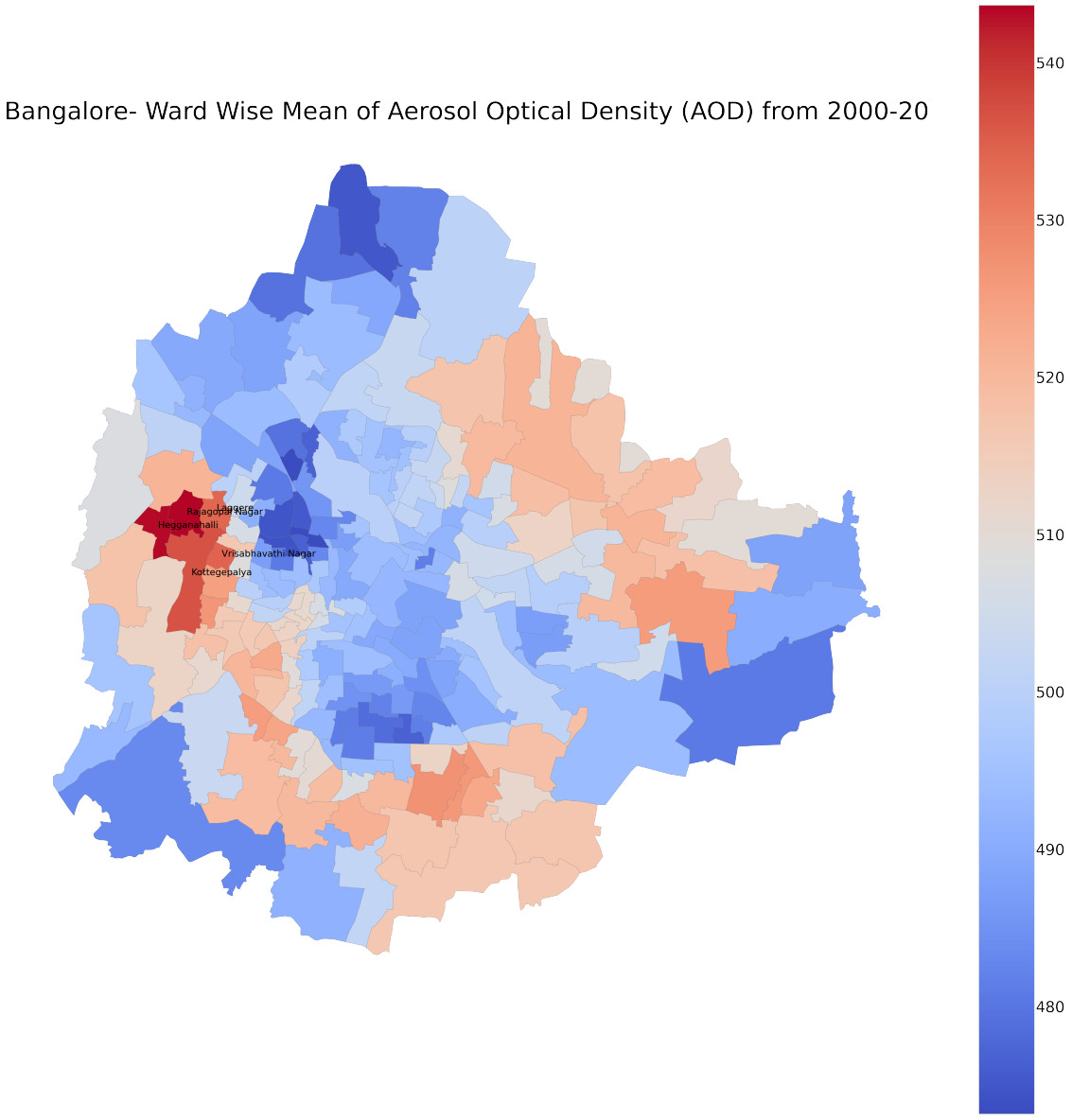
```
sm = plt.cm.ScalarMappable(cmap='coolwarm',norm=plt.
 ↪Normalize(vmin=vmin,vmax=vmax ))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_from_2011_AOD.plot(column=variable, cmap='coolwarm',␣
 ↪linewidth=0.8, ax=ax,edgecolor='0.5',
                                          figsize = (150,450))
count= 0
for x, y, label in zip(merged_Bangalore_from_2011_AOD.centroid.x,␣
 ↪merged_Bangalore_from_2011_AOD.centroid.y,
                       merged_Bangalore_from_2011_AOD.WARD_NAME ):
    if label not in lst:
        continue
    count = count + 1
#     print(x,y,label)
    ax1.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 35)
plt.show()
```



The chloropleth plotted here shows the spread of the AOD values in the wards of Bangalore for the time period of 2011-2020. It labels the Wards where there has been an increment of over 25 % in the AOD Values.

[33]:
```
# For 2001-20
lst1 = AOD_wards.sort_values(by = ["AOD"], ascending = False).head(5).index.
 ↪tolist()
lst1 = [Bangalore_Wards_List.loc[i][0] for i in lst1]

# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
```

```
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Aerosol Optical Density (AOD) from␣
 ↪2000-20',fontdict={'fontsize': '120', 'fontweight' : '3'})

sm = plt.cm.ScalarMappable(cmap='coolwarm',norm=plt.
 ↪Normalize(vmin=vmin,vmax=vmax ))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_AOD.plot(column=variable, cmap='coolwarm', linewidth=0.
 ↪8, ax=ax,edgecolor='0.5', figsize = (150,450))

count= 0
for x, y, label in zip(merged_Bangalore_AOD.centroid.x, merged_Bangalore_AOD.
 ↪centroid.y,
                       merged_Bangalore_AOD.WARD_NAME ):
    if label not in lst1:
        continue
    count = count + 1
    if count % 3 == 0:
#     print(x,y,label)
        ax1.annotate(label, xy=(x, y+0.00001), xytext=(3, 3),␣
 ↪textcoords="offset points",fontsize = 45)
        continue
    ax1.annotate(label, xy=(x, y-0.00001), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 45)
plt.show()
print("Wards with Highest AOD Values are :",lst1)
```




```
Wards with Highest AOD Values are : ['Rajagopal Nagar', 'Hegganahalli',
'Kottegepalya', 'Vrisabhavathi Nagar', 'Laggere']
```

The chloropleth shows the wards where there has been a high rise in the AOD values in the last 20 years

Variable 2 : Temperature (min_temp at 2 m, mean_temp and max_temp at 2 m) - ERA Dataset

Temperature is a physical quantity that expresses hot and cold. It is the manifestation of thermal energy, present in all matter, which is the source of the occurrence of heat, a flow of energy, when a body is in contact with another that is colder. Temperature is measured with a thermometer. Its SI Unit is Kelvin.

[34]:
```
ERA = pd.read_csv('Bangalore/ERA5_T.csv')
ERA.date = ERA.date.str[:10]
ERA.date = pd.to_datetime(ERA.date)
ERA =␣
 ↪ERA[["WARD_NO","date","maximum_2m_air_temperature","minimum_2m_air_temperature"]]
ERA_MEAN = pd.read_csv("Bangalore/ERA_TMEAN.csv")
ERA_MEAN.date = ERA_MEAN.date.str[:10]
ERA_MEAN.date = pd.to_datetime(ERA_MEAN.date)
ERA = ERA.merge(ERA_MEAN,how = "inner" , left_on = ["WARD_NO","date"] ,␣
 ↪right_on = ["WARD_NO","date"])
ERA
```

[34]:

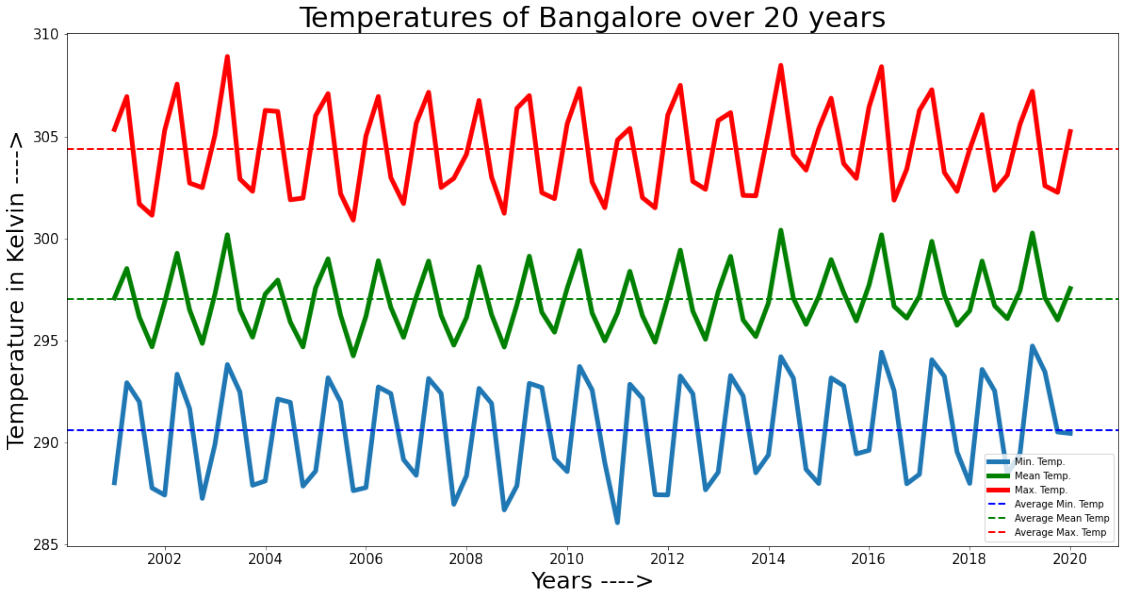
	WARD_NO	date	maximum_2m_air_temperature	minimum_2m_air_temperature	Temperature_Mean
0	168	2001-01-01	305.332336	288.126648	297.093414
1	169	2001-01-01	305.332336	288.126648	297.093414
2	179	2001-01-01	305.332336	288.126648	297.093414
3	149	2001-01-01	305.295791	288.078343	296.993614
4	150	2001-01-01	305.332336	288.126648	297.093414
…	…	…	…	…	…
15241	151	2020-01-01	305.183746	290.525635	297.566040
15242	152	2020-01-01	305.183746	290.525635	297.566040
15243	178	2020-01-01	305.183746	290.525635	297.566040
15244	173	2020-01-01	305.183746	290.525635	297.566040
15245	176	2020-01-01	305.183746	290.525635	297.566040

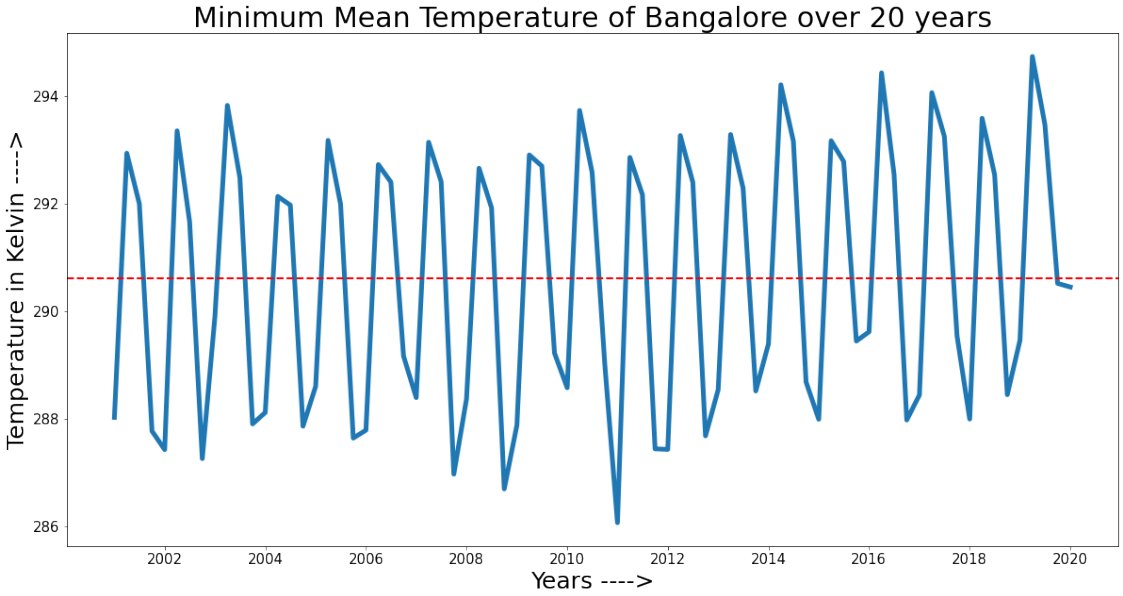
[15246 rows x 5 columns]

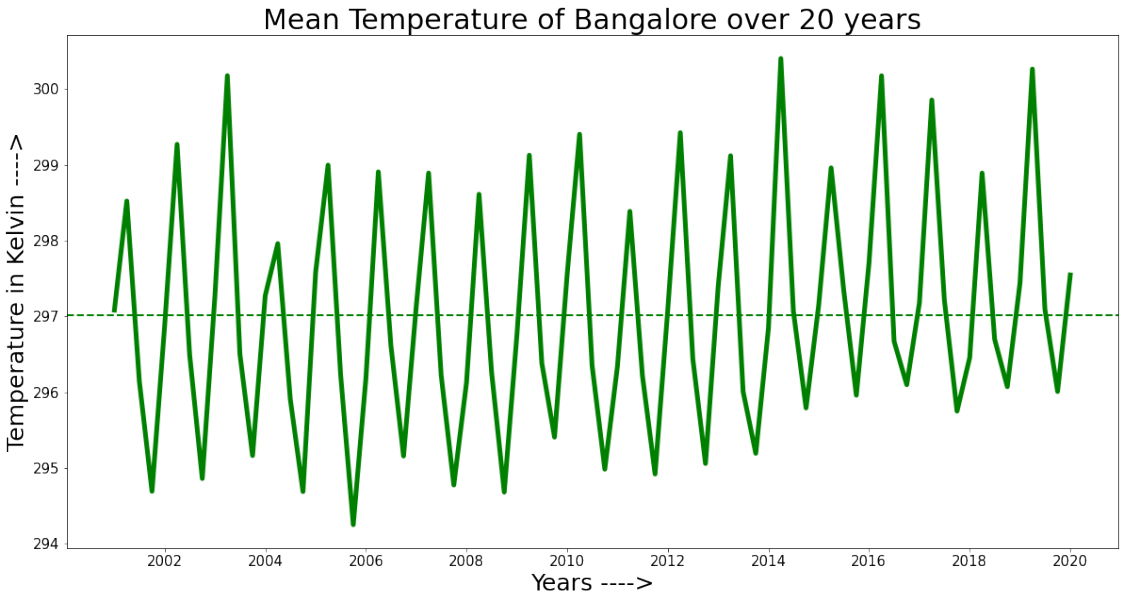
[35]:
```
# Plotting
ERA_date = ERA.groupby("date").mean()

plt.figure(figsize = (20,10))
plt.plot(ERA_date.index,ERA_date.minimum_2m_air_temperature , lw = 5, label =␣
 ↪"Min. Temp.")
```

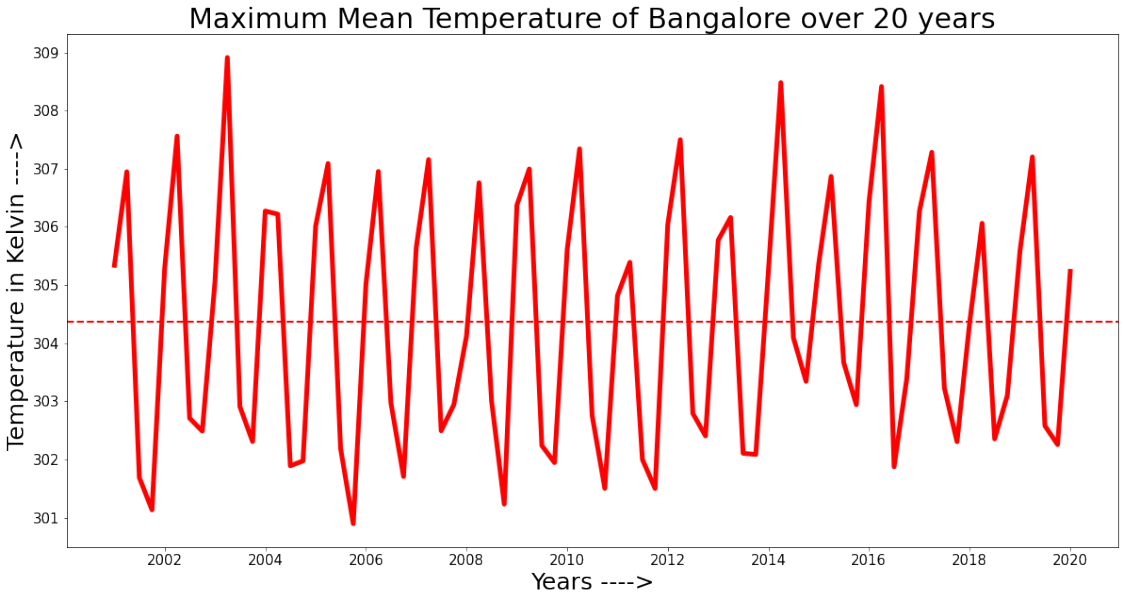
```
plt.plot(ERA_date.index,ERA_date.Temperature_Mean , lw = 5, color = "g", label␣
↪= "Mean Temp.")
plt.plot(ERA_date.index,ERA_date.maximum_2m_air_temperature , lw = 5, color =␣
↪"red", label = "Max. Temp.")
plt.title("Temperatures of Bangalore over 20 years", fontsize = 30)
plt.ylabel("Temperature in Kelvin ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(ERA_date["minimum_2m_air_temperature"].mean(), color = "blue", lw =␣
↪2, linestyle = "dashed",
            label = "Average Min. Temp")
plt.axhline(ERA_date["Temperature_Mean"].mean(), color = "green", lw = 2,␣
↪linestyle = "dashed",
           label = "Average Mean Temp")
plt.axhline(ERA_date["maximum_2m_air_temperature"].mean(), color = "red", lw =␣
↪2, linestyle = "dashed",
           label = "Average Max. Temp")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.legend()
plt.show()

plt.figure(figsize = (20,10))
plt.plot(ERA_date.index,ERA_date["minimum_2m_air_temperature"] , lw = 5)
plt.title("Minimum Mean Temperature of Bangalore over 20 years", fontsize = 30)
plt.ylabel("Temperature in Kelvin ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(ERA_date["minimum_2m_air_temperature"].mean(), color = "r", lw = 2,␣
↪linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()

plt.figure(figsize = (20,10))
plt.plot(ERA_date.index,ERA_date.Temperature_Mean , lw = 5, color = "green")
plt.title("Mean Temperature of Bangalore over 20 years", fontsize = 30)
plt.ylabel("Temperature in Kelvin ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(ERA_date["Temperature_Mean"].mean(), color = "green", lw = 2,␣
↪linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()

plt.figure(figsize = (20,10))
plt.plot(ERA_date.index,ERA_date.maximum_2m_air_temperature , lw = 5, color =␣
↪"r")
```

```
plt.title("Maximum Mean Temperature of Bangalore over 20 years", fontsize = 30)
plt.ylabel("Temperature in Kelvin ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(ERA_date["maximum_2m_air_temperature"].mean(), color = "r", lw = 2,␣
 ↪linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()
```









The above graphs show the Minimum, Mean and Maximum temperatures of Bangalore over the last 20 years.

[36]:
```
ERA_till_2010 = ERA[(ERA["date"]>= "20010101") & (ERA["date"]< "20110101")]
ERA_till_2010 = ERA_till_2010.groupby("WARD_NO").mean()
ERA_from_2011 = ERA[(ERA["date"]>= "20110101")]
ERA_from_2011 = ERA_from_2011.groupby("WARD_NO").mean()
```

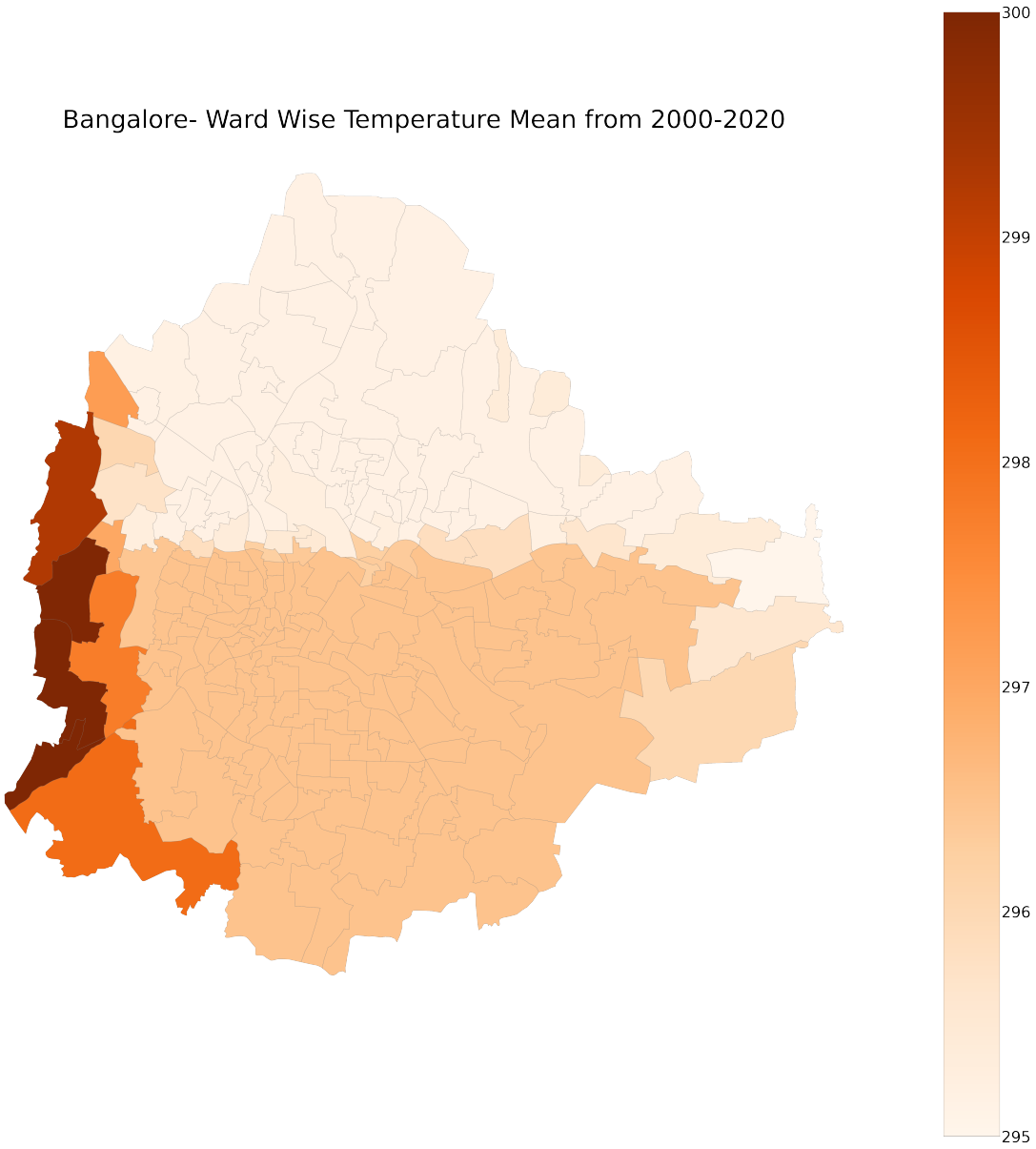
```
ERA_wards = ERA.groupby("WARD_NO").mean()
```

[37]:
```
merged_Bangalore_till_2010_ERA = map_Bangalore.merge(ERA_till_2010, left_on =␣
↪"WARD_NO",right_index = True)
merged_Bangalore_from_2011_ERA = map_Bangalore.merge(ERA_from_2011, left_on =␣
↪"WARD_NO",right_index = True)
merged_Bangalore_ERA           = map_Bangalore.merge(ERA_wards, left_on =␣
↪"WARD_NO",right_index = True)
```

[38]:
```
# counting wards who have temperature dfference by 0.45 Kelvin or more in last␣
↪decade
ward_list = Bangalore_Wards_List["Ward_Name"].tolist()
ERA_worse_wards = ((ERA_from_2011["Temperature_Mean"] -␣
↪ERA_till_2010["Temperature_Mean"])/ERA_till_2010["Temperature_Mean"] ).
↪tolist()
lst = []
lst1 = []
for i in range(len(ERA_worse_wards)):
    # print(i)
    if((ERA_worse_wards[i] >= 0.0015) | (ERA_worse_wards[i] <= -0.0015 )):
        lst.append((ward_list[i]))

# print(lst)
```

[39]:
```
# Plotting 2000-20
variable = 'Temperature_Mean'
# # set the range for the choropleth values
vmin, vmax = 295,300 # merged_Bangalore_ERA["Temperature_Mean"].min(),␣
↪merged_Bangalore_ERA["Temperature_Mean"].max()
# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Temperature Mean from␣
↪2000-2020',fontdict={'fontsize': '120', 'fontweight' : '3'})

sm = plt.cm.ScalarMappable(cmap='Oranges',norm=plt.
↪Normalize(vmin=vmin,vmax=vmax ))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
```

```
ax1 = merged_Bangalore_ERA.plot(column=variable, cmap='Oranges', linewidth=0.8,␣
 ↪ax=ax,edgecolor='0.5', figsize = (150,450))

# count= 0
# for x, y, label in zip(merged_Bangalore_from_2011.centroid.x,␣
 ↪merged_Bangalore_from_2011.centroid.y,
#                        merged_Bangalore_from_2011.WARD_NAME ):
#     if label not in lst:
#         continue
#     count = count + 1
# #     print(x,y,label)
#     ax1.annotate(label, xy=(x+0.01, y+0.01), xytext=(3, 3),␣
 ↪textcoords="offset points",fontsize = 35)
plt.show()
```



The chloropleth shows the mean temperature of the wards of Bangalore. The northern part of the Bangalore is colder than the southern part due to the prescence of hilly terrain in the northern part.

Variable 3 : Evapotranspiration - MODIS DATASET

It is the sum of evaporation and plant transpiration from the Earth's land and ocean surface to the atmosphere. Evaporation accounts for the movement of water to the air from sources such as the soil, canopy interception, and water bodies.

[40]:
```
# Loading
EvapoTranspiration = pd.read_csv('Bangalore/MODIS_EvapoTrans.csv')
```

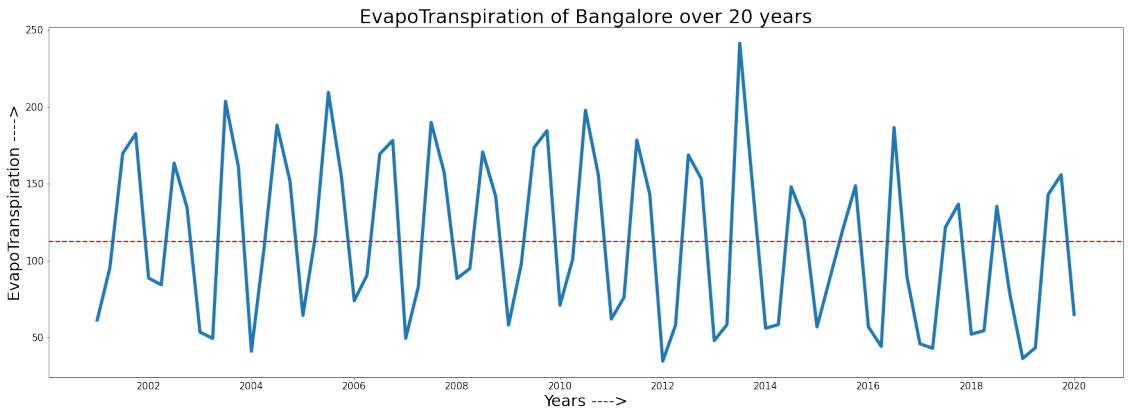
```
EvapoTranspiration.date = pd.to_datetime(EvapoTranspiration.date)
EvapoTranspiration.rename(columns={'mean':'EvapoTranspiration'}, inplace=True)
EvapoTranspiration
```

[40]:

	WARD_NO	date	EvapoTranspiration
0	2	2001-01-01	61.913493
1	3	2001-01-01	59.552162
2	4	2001-01-01	64.469620
3	53	2001-01-01	55.868946
4	54	2001-01-01	60.452495
…	…	…	…
7077	177	2020-01-01	47.778448
7078	26	2020-01-01	57.368084
7079	25	2020-01-01	60.905588
7080	86	2020-01-01	61.838838
7081	198	2020-01-01	70.114378

[7082 rows x 3 columns]

[41]:
```
%matplotlib inline
plt.figure(figsize = (30,10))
EvapoTranspiration_date = EvapoTranspiration.groupby("date").mean()
# EvapoTranspiration_date.head(50)
# plt.scatter(EvapoTranspiration_date.index,EvapoTranspiration_date.
 ↪EvapoTranspiration)

plt.plot(EvapoTranspiration_date.index.
 ↪tolist(),EvapoTranspiration_date["EvapoTranspiration"] , lw = 5)
plt.title("EvapoTranspiration of Bangalore over 20 years", fontsize = 30)
plt.ylabel("EvapoTranspiration ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(EvapoTranspiration_date["EvapoTranspiration"].mean(), color = "r",␣
 ↪lw = 2, linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()
```



The graph shows the trends of the EvapoTranspiration over the last 20 years in Bangalore

[42]:
```
EvapoTranspiration_till_2010 = EvapoTranspiration[(EvapoTranspiration["date"]>=␣
↪"20010101") & (EvapoTranspiration["date"]< "20110101")]
EvapoTranspiration_till_2010 = EvapoTranspiration_till_2010.groupby("WARD_NO").
↪mean()
EvapoTranspiration_from_2011 = EvapoTranspiration[(EvapoTranspiration["date"]>=␣
↪"20110101")]
EvapoTranspiration_from_2011 = EvapoTranspiration_from_2011.groupby("WARD_NO").
↪mean()
EvapoTranspiration_wards = EvapoTranspiration.groupby("WARD_NO").mean()
```

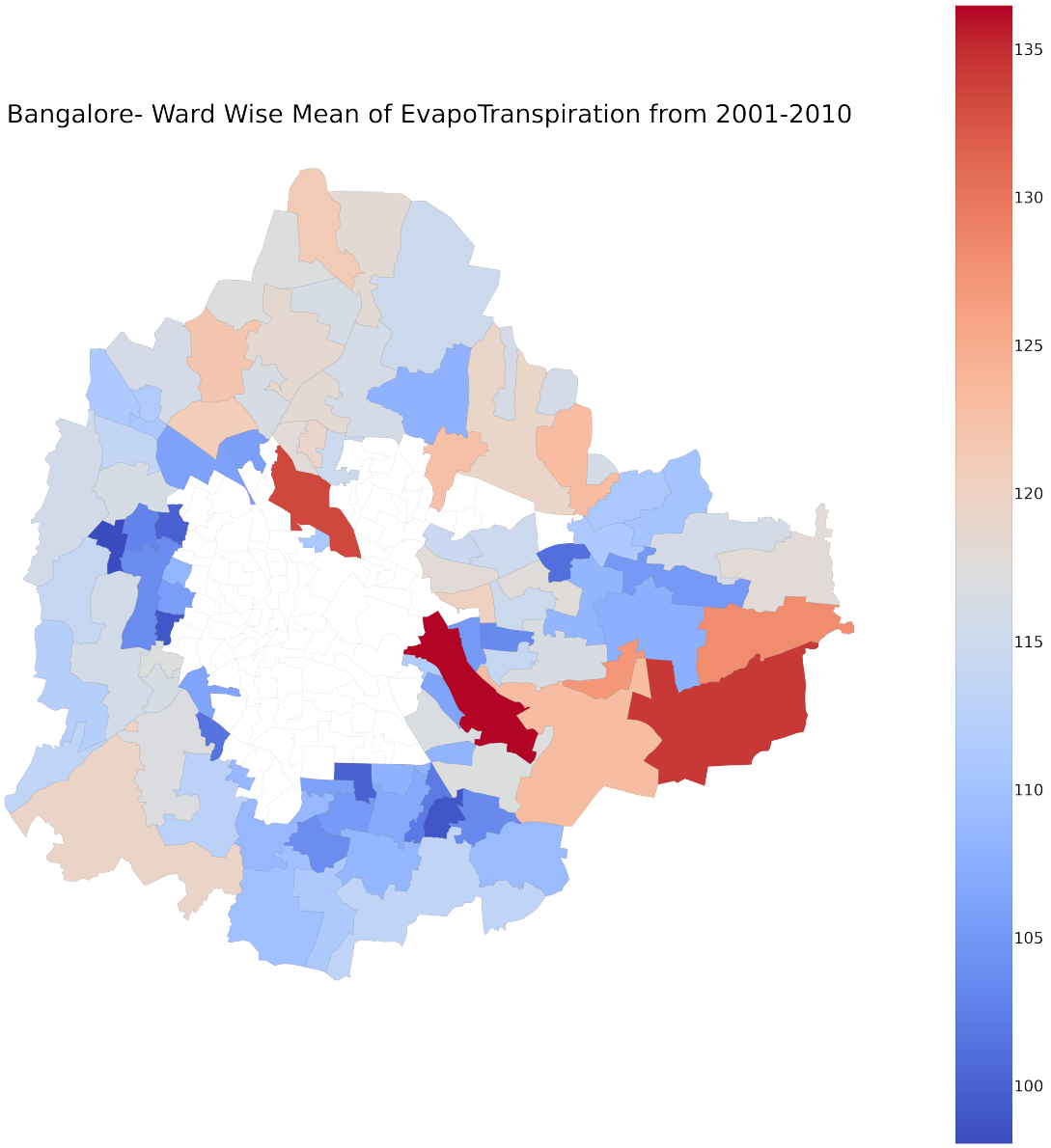
[43]:
```
merged_Bangalore_till_2010_EvapoTranspiration = map_Bangalore.
↪merge(EvapoTranspiration_till_2010, how = "left", left_on =␣
↪"WARD_NO",right_index = True)
merged_Bangalore_from_2011_EvapoTranspiration = map_Bangalore.
↪merge(EvapoTranspiration_from_2011, how = "left",left_on =␣
↪"WARD_NO",right_index = True)
merged_Bangalore_EvapoTranspiration       = map_Bangalore.
↪merge(EvapoTranspiration_wards,how = "left", left_on = "WARD_NO",right_index␣
↪= True)
```

[44]:
```
# Finding Wards where Evapotranspiration has decreased by 25 % from last decade
EvapoTranspiration_worse_wards =␣
↪((EvapoTranspiration_from_2011["EvapoTranspiration"] -␣
↪EvapoTranspiration_till_2010["EvapoTranspiration"])/
↪EvapoTranspiration_till_2010["EvapoTranspiration"] ).tolist()
lst = []
lst1 = []
for i in range(len(EvapoTranspiration_worse_wards)):
    if EvapoTranspiration_worse_wards[i] < -0.25:
        lst.append(ward_list[i])
```

```
        lst1.append(i)
print("These are the Wards which have observed decrease in evapotranspiration␣
 ↪in the last decade by at least 25%\n",lst)
```

```
These are the Wards which have observed decrease in evapotranspiration in the
last decade by at least 25%
 ['T Dasarahalli', 'Hebbala', 'Nagavara', 'Ramamurthy Nagar', 'Kammanahalli',
'Gangenahalli', 'Chokkasandra', 'Dodda Bidarakallu', 'Peenya Industrial Area',
'Vijnanapura', 'Hudi', 'Nagapura', 'Laggere', 'Kadugodi', 'Hagadur']
```

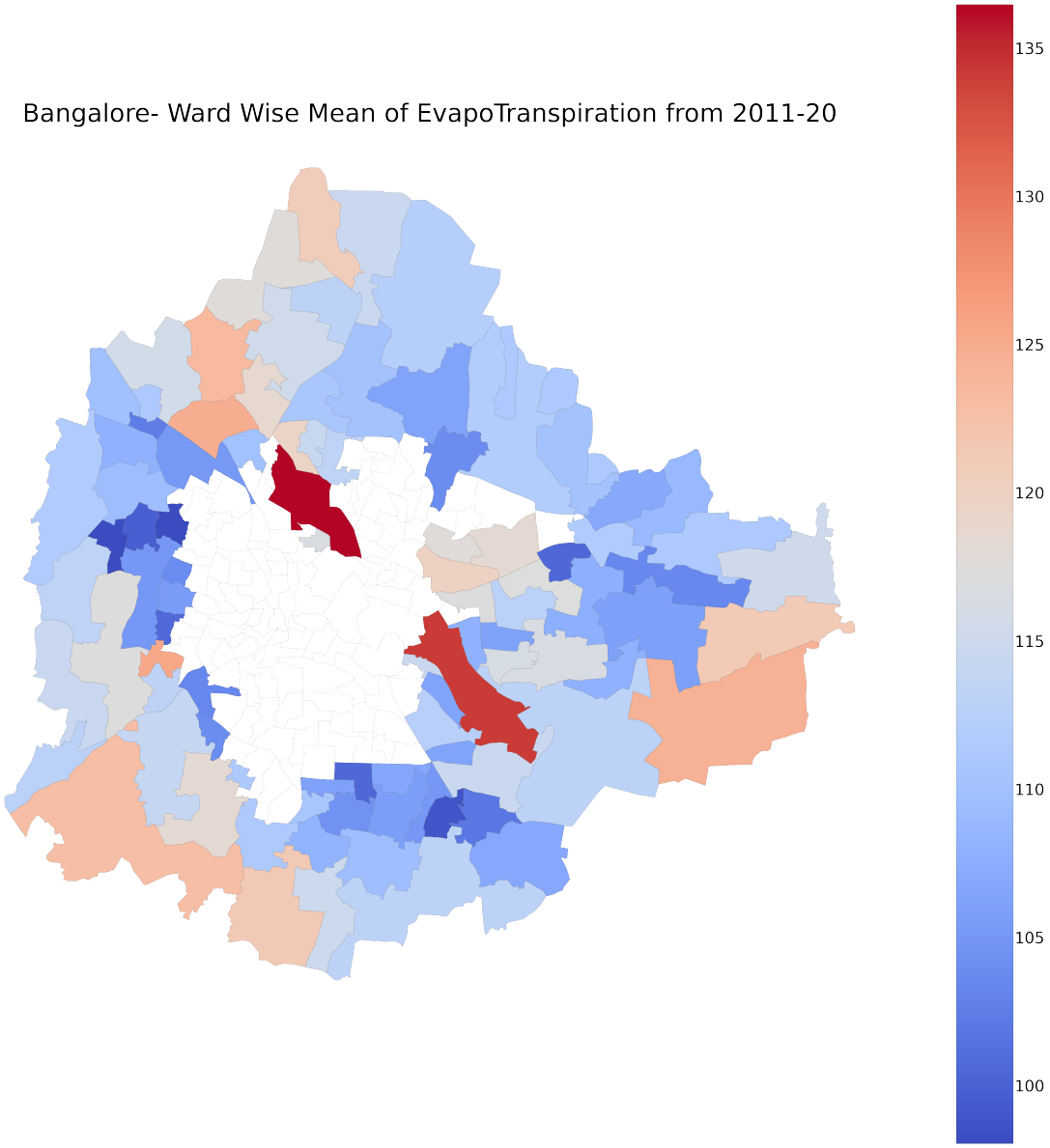
[45]:
```
# Plotting 2001-10

# EvapoTranspiration_wards_1 = EvapoTranspiration.groupby("WARD_NO").mean()
#␣
 ↪print(len(EvapoTranspiration_till_2010_EvapoTranspiration),len(EvapoTranspiration_from_2011
# fig, (ax1,ax2) = plt.subplots(1,2,figsize = (8,8), sharex = True, sharey=True)
variable = 'EvapoTranspiration'
# # set the range for the choropleth values
vmin, vmax = merged_Bangalore_EvapoTranspiration["EvapoTranspiration"].min(),␣
 ↪merged_Bangalore_EvapoTranspiration["EvapoTranspiration"].max()
# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of EvapoTranspiration from 2001-2010',
             fontdict={'fontsize': '120', 'fontweight' : '3'})


sm = plt.cm.ScalarMappable(cmap='coolwarm',norm=plt.Normalize(vmin=vmin
                                                              ,vmax=vmax ))
# empty array for the data range
sm.set_array([])

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
merged_Bangalore_till_2010_EvapoTranspiration.plot(column=variable,␣
 ↪cmap='coolwarm', linewidth=0.8, ax=ax,
                                edgecolor='0.5', figsize = (150,450))
plt.show()
```



The chloropleth show the ward wise mean evapotranspiration for the time period of 2001-2010

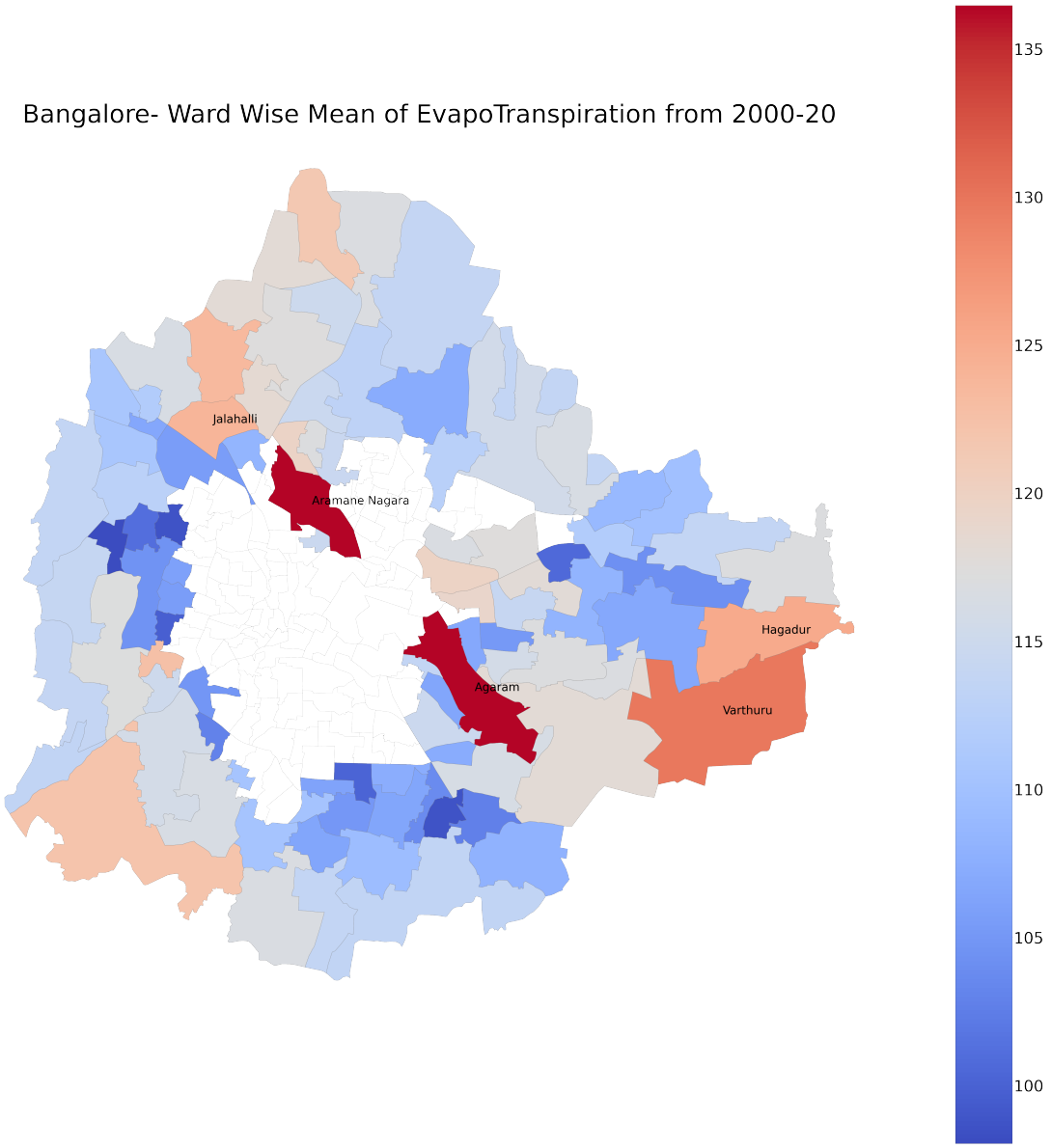
[46]:
```
# Plotting 2011-20
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of EvapoTranspiration from 2011-20'
             ,fontdict={'fontsize': '120', 'fontweight' : '3'})
```

```
sm = plt.cm.ScalarMappable(cmap='coolwarm',norm=plt.
 ↪Normalize(vmin=vmin,vmax=vmax ))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_from_2011_EvapoTranspiration.plot(column=variable,␣
 ↪cmap='coolwarm', linewidth=0.8, ax=ax,edgecolor='0.5',
                                          figsize = (150,450))
plt.show()
```



The chloropleth show the ward wise mean evapotranspiration for the time period of 2011-2020

[47]:
```
# Plotting 2000-20
lst1 = EvapoTranspiration_wards.sort_values(by = ["EvapoTranspiration"],␣
 ↪ascending = False).head(5).index.tolist()
lst1 = [Bangalore_Wards_List.loc[i][0] for i in lst1]

# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of EvapoTranspiration from␣
 ↪2000-20',fontdict={'fontsize': '120', 'fontweight' : '3'})

sm = plt.cm.ScalarMappable(cmap='coolwarm',norm=plt.
 ↪Normalize(vmin=vmin,vmax=vmax ))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_EvapoTranspiration.plot(column=variable,␣
 ↪cmap='coolwarm', linewidth=0.8, ax=ax,edgecolor='0.5', figsize = (150,450))

count= 0
for x, y, label in zip(merged_Bangalore_EvapoTranspiration.centroid.x,␣
 ↪merged_Bangalore_EvapoTranspiration.centroid.y,
                       merged_Bangalore_EvapoTranspiration.WARD_NAME ):
    if label not in lst1:
        continue
    count = count + 1
    if count % 3 == 0:
#     print(x,y,label)
        ax1.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 55)
        continue
    ax1.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 55)
plt.show()
print("The five wards with Maximum Evapotranspiration are :",lst1)
```




```
The five wards with Maximum Evapotranspiration are : ['Agaram', 'Aramane
Nagara', 'Varthuru', 'Hagadur', 'Jalahalli']
```

Variable 4: Green Vegetation Measure with NDVI and EVI mean indices - MODIS Dataset

The NDVI is a dimensionless index that describes the difference between visible and near-infrared reflectance of vegetation cover and can be used to estimate the density of green on an area of land (Weier and Herring, 2000).

EVI is similar to Normalized Difference Vegetation Index (NDVI) and can be used to quantify vegetation greenness. However, EVI corrects for some atmospheric conditions and canopy background noise and is more sensitive in areas with dense vegetation.

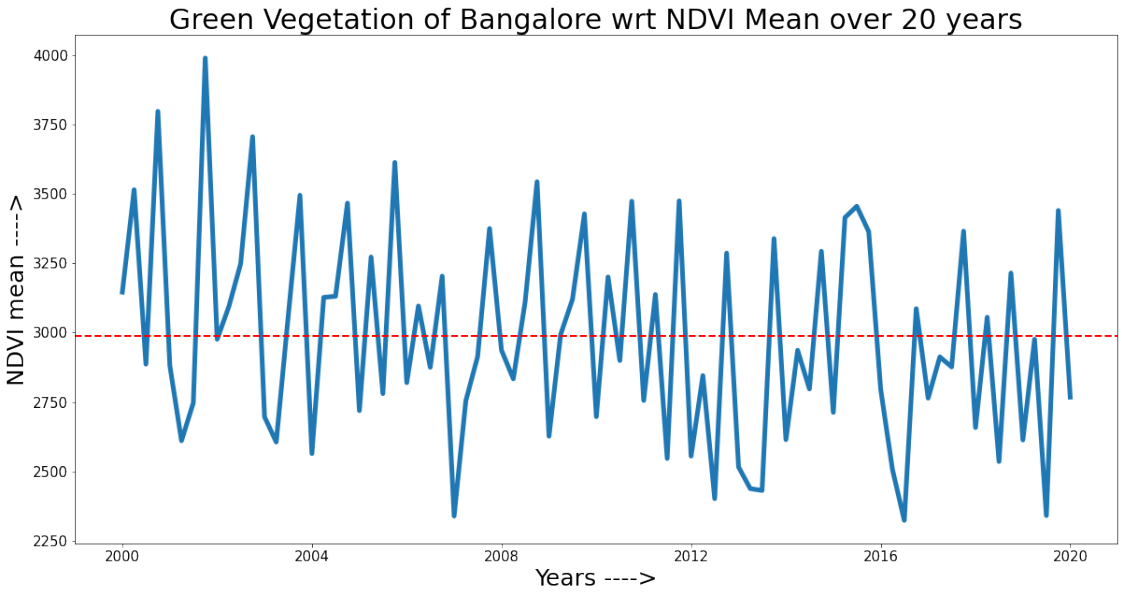
[48]:
```
Green_Vegetation = pd.read_csv('Bangalore/MODIS_Green_Q.csv')
Green_Vegetation.date = pd.to_datetime(Green_Vegetation.date)
Green_Vegetation = Green_Vegetation[["WARD_NO","date","EVI_mean","NDVI_mean"]]
Green_Vegetation
```

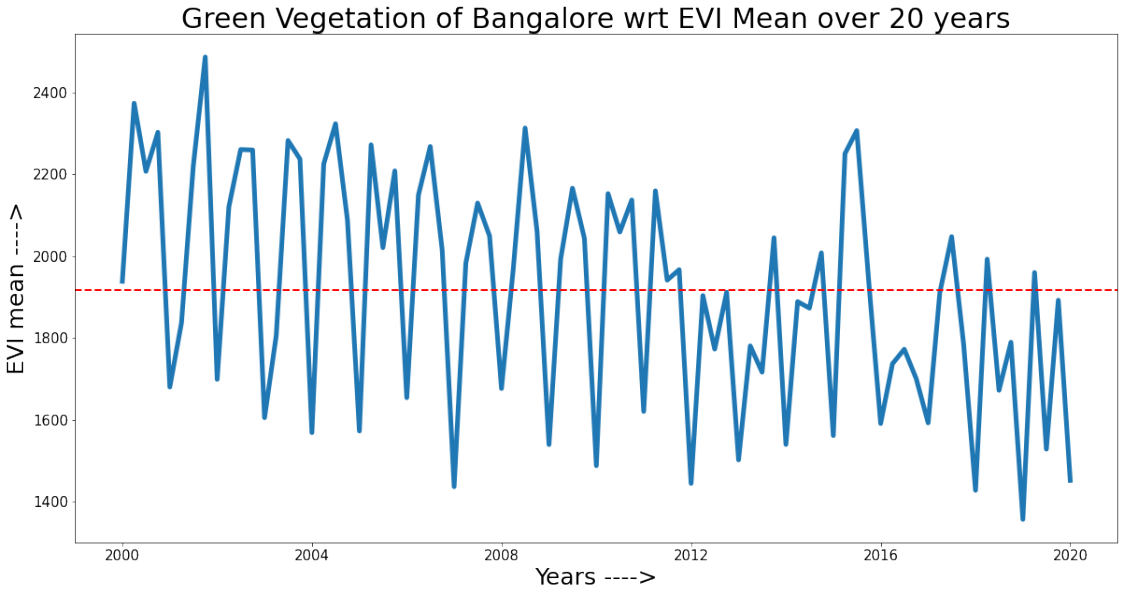
[48]:

	WARD_NO	date	EVI_mean	NDVI_mean
0	2	2000-01-01	2310.362522	3976.583789
1	3	2000-01-01	2185.881533	3665.445875
2	4	2000-01-01	2327.046428	3961.514236
3	51	2000-01-01	1822.855997	2933.833973
4	53	2000-01-01	1761.592954	2966.032267
…	…	…	…	…
16033	172	2020-01-01	1232.606843	2335.401736
16034	26	2020-01-01	1597.486161	2816.833160
16035	25	2020-01-01	1540.020164	2753.535720
16036	86	2020-01-01	1610.859884	2862.603246
16037	198	2020-01-01	1928.336195	3635.664204

[16038 rows x 4 columns]

[49]:
```
Green_Vegetation_date = Green_Vegetation.groupby("date").mean()
plt.figure(figsize = (20,10))
plt.plot(Green_Vegetation_date.index,Green_Vegetation_date["NDVI_mean"] , lw =␣
↪5)
plt.title("Green Vegetation of Bangalore wrt NDVI Mean over 20 years", fontsize␣
↪= 30)
plt.ylabel("NDVI mean ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(Green_Vegetation_date["NDVI_mean"].mean(), color = "r", lw = 2,␣
↪linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()

plt.figure(figsize = (20,10))
plt.plot(Green_Vegetation_date.index,Green_Vegetation_date["EVI_mean"] , lw = 5)
plt.title("Green Vegetation of Bangalore wrt EVI Mean over 20 years", fontsize␣
↪= 30)
plt.ylabel("EVI mean ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(Green_Vegetation_date["EVI_mean"].mean(), color = "r", lw = 2,␣
↪linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()
```

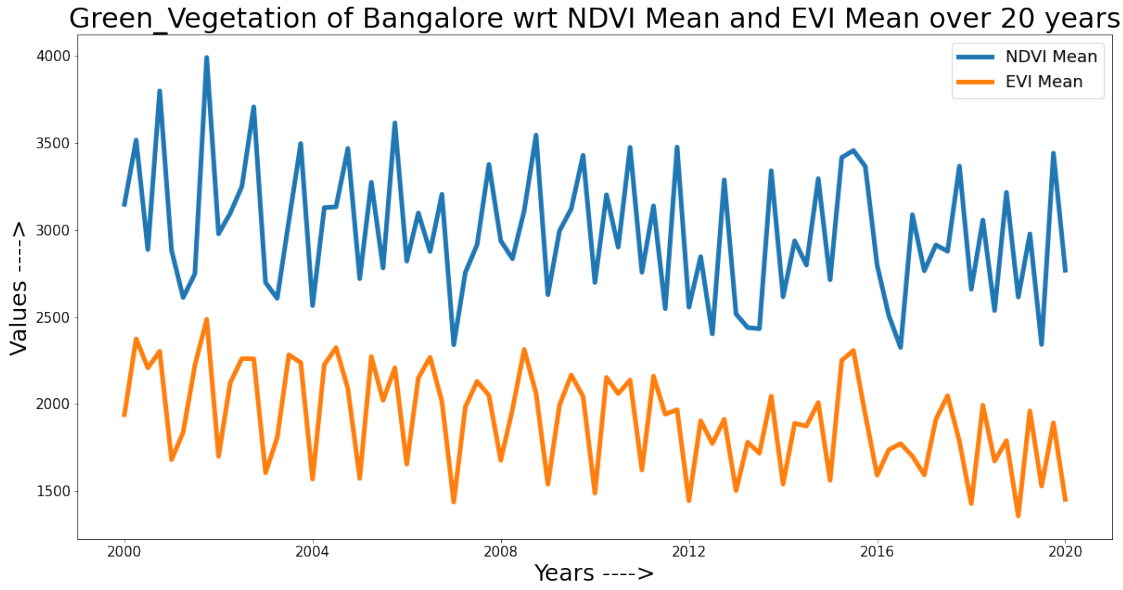




The plots above show the NVDI and the EVI indices of Bangalore over the last 20 years. NDVI and EVI are the measures of the green vegetation of a region.

[50]:
```
Green_Vegetation_date = Green_Vegetation.groupby("date").mean()
plt.figure(figsize = (20,10))
plt.plot(Green_Vegetation_date.index,Green_Vegetation_date["NDVI_mean"], lw =␣
↪5, label = "NDVI Mean")
```

```
plt.plot(Green_Vegetation_date.index,Green_Vegetation_date["EVI_mean"] , lw = 5␣
 ↪, label = "EVI Mean")
plt.title("Green_Vegetation of Bangalore wrt NDVI Mean and EVI Mean over 20␣
 ↪years", fontsize = 30)
plt.ylabel("Values ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.legend(fontsize = 18, loc = 1)
plt.show()
```



Depicts Variation in NDVI and EVI for Bangalore over time

[51]:
```
Green_Vegetation_till_2010 = Green_Vegetation[(Green_Vegetation["date"]>=␣
 ↪"20010101") & (Green_Vegetation["date"]< "20110101")]
Green_Vegetation_till_2010 = Green_Vegetation_till_2010.groupby("WARD_NO").
 ↪mean()
Green_Vegetation_from_2011 = Green_Vegetation[(Green_Vegetation["date"]>=␣
 ↪"20110101")]
Green_Vegetation_from_2011 = Green_Vegetation_from_2011.groupby("WARD_NO").
 ↪mean()
Green_Vegetation_wards = Green_Vegetation.groupby("WARD_NO").mean()
```

[52]:
```
merged_Bangalore_till_2010_Green_Vegetation = map_Bangalore.
 ↪merge(Green_Vegetation_till_2010, left_on = "WARD_NO",right_index = True)
merged_Bangalore_from_2011_Green_Vegetation = map_Bangalore.
 ↪merge(Green_Vegetation_from_2011, left_on = "WARD_NO",right_index = True)
```

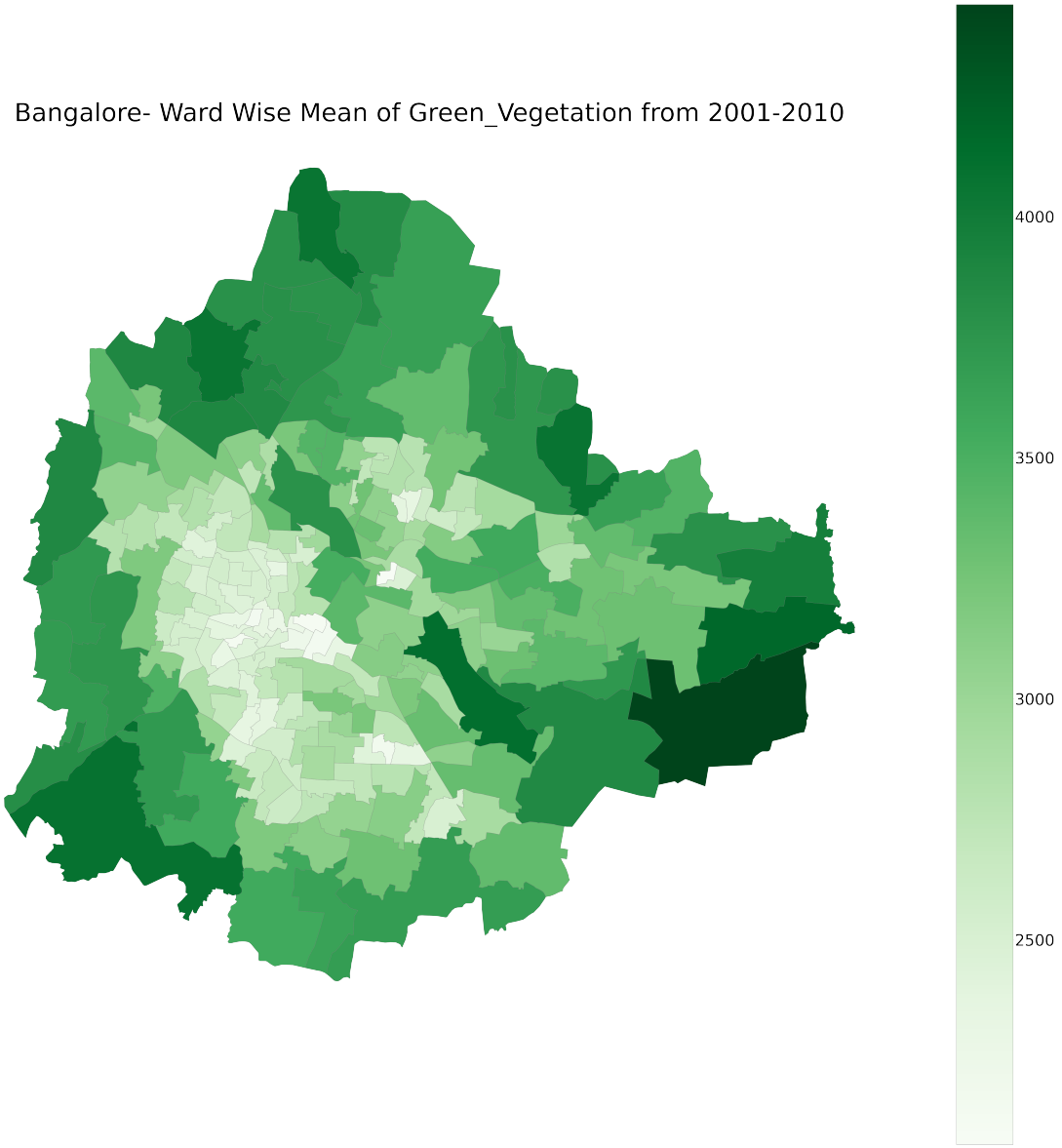
```
merged_Bangalore_Green_Vegetation       = map_Bangalore.
 ↪merge(Green_Vegetation_wards, left_on = "WARD_NO",right_index = True)
```

[53]:
```
# Finding Wards who have 15 % decrease in Green Vegetation in Past Decade
Green_Vegetation_worse_wards = ((Green_Vegetation_from_2011["NDVI_mean"] -␣
 ↪Green_Vegetation_till_2010["NDVI_mean"])/
 ↪Green_Vegetation_till_2010["NDVI_mean"] ).tolist()
lst = []
lst1 = []
for i in range(len(Green_Vegetation_worse_wards)):
    if(Green_Vegetation_worse_wards[i] < -0.15):
        lst.append((Bangalore_Wards_List.loc[i][0]))
        lst1.append(i)
        # print(Bangalore_Wards_List.loc[i][0])
print("These are the wards which show 15% reduction in the Vegetation cover in␣
 ↪the last decade", lst )
```

```
These are the wards which show 15% reduction in the Vegetation cover in the last
decade ['Horamavu', 'Mahalakshimpuram', 'Rajagopal Nagar', 'Dodda Nekkundi',
'HSR Layout', 'Bilekhalli']
```

[54]:
```
# 2001-10

# Green_Vegetation_wards_1 = Green_Vegetation.groupby("WARD_NO").mean()
#␣
 ↪print(len(Green_Vegetation_till_2010_Green_Vegetation),len(Green_Vegetation_from_2011_Green
# fig, (ax1,ax2) = plt.subplots(1,2,figsize = (8,8), sharex = True, sharey=True)
variable = 'NDVI_mean'
# # set the range for the choropleth values
vmin, vmax = merged_Bangalore_Green_Vegetation["NDVI_mean"].min(),␣
 ↪merged_Bangalore_Green_Vegetation["NDVI_mean"].max()
# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Green_Vegetation from 2001-2010',
             fontdict={'fontsize': '120', 'fontweight' : '3'})


sm = plt.cm.ScalarMappable(cmap='Greens',norm=plt.Normalize(vmin=vmin
                                                            ,vmax=vmax ))
# empty array for the data range
sm.set_array([])

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
```

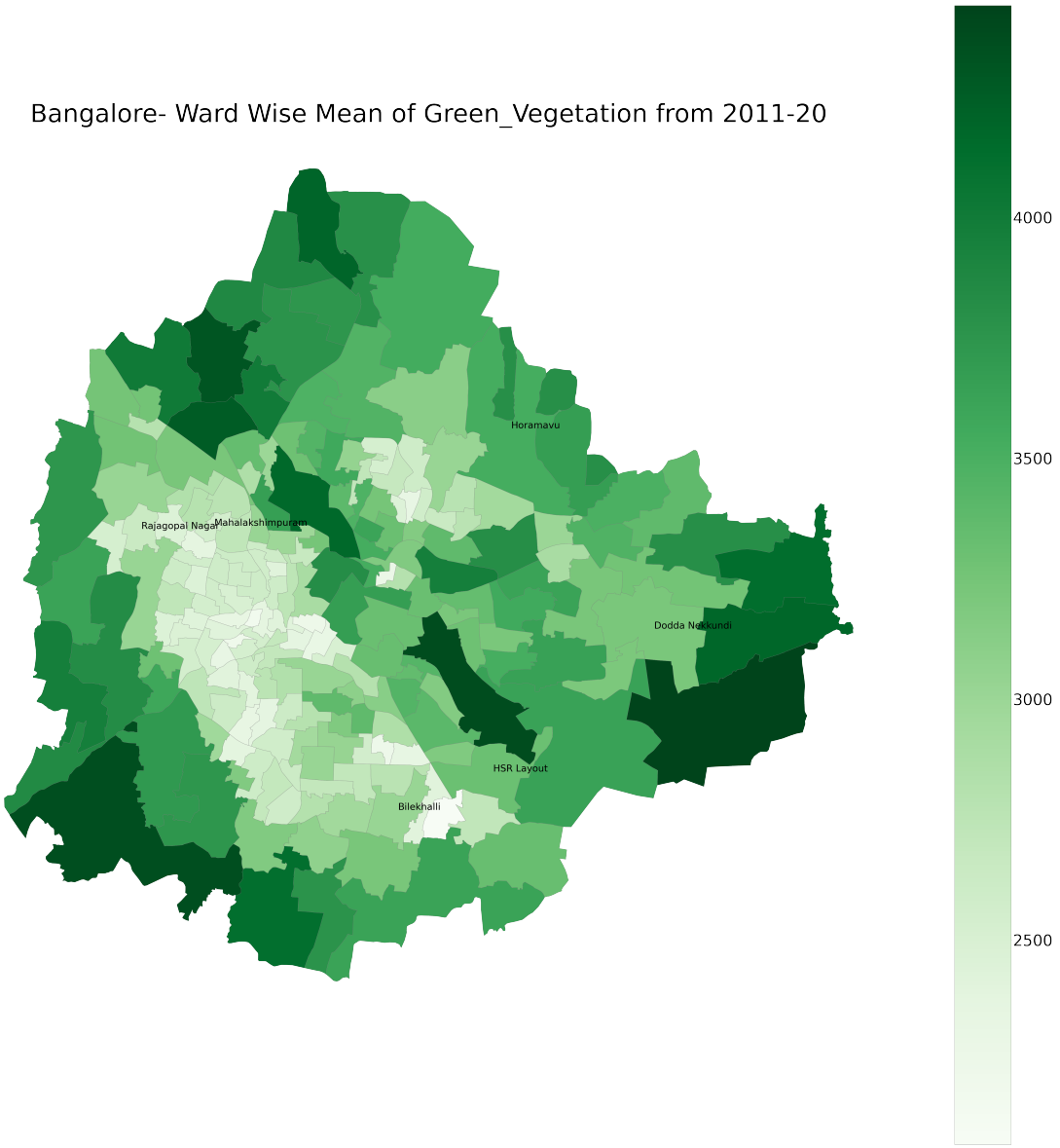
```
# create map
merged_Bangalore_till_2010_Green_Vegetation.plot(column=variable,␣
 ↪cmap='Greens', linewidth=0.8, ax=ax,
                                  edgecolor='0.5', figsize = (150,450))
plt.show()
```



The chloropleth above show the mean NDVI mean of the wards of Banglorw over the time period of 2001-2010

[55]:
```
# Plotting 2011-20

fig, ax = plt.subplots(1, figsize=(100, 100))
```

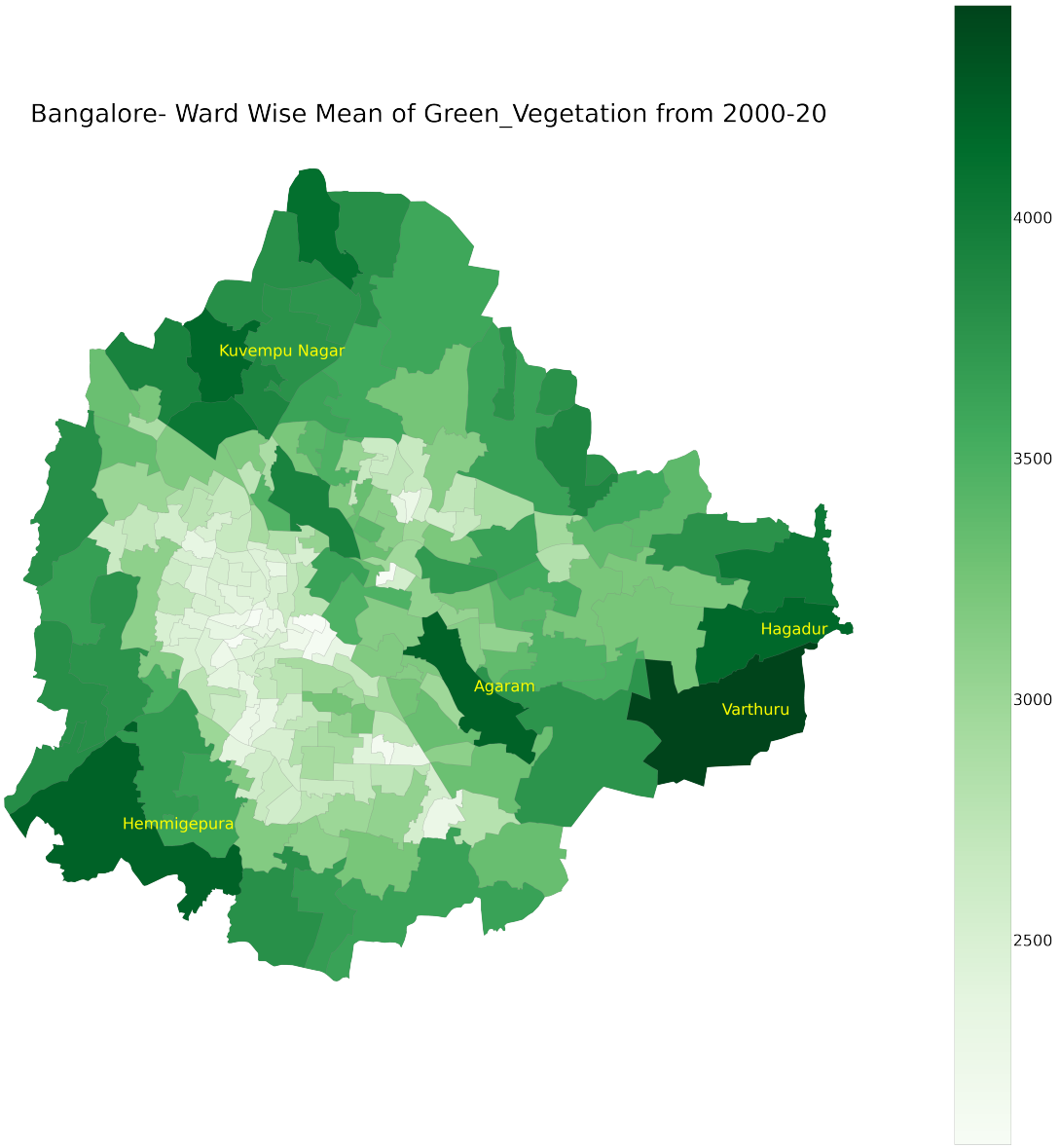
```
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Green_Vegetation from 2011-20'
             ,fontdict={'fontsize': '120', 'fontweight' : '3'})

sm = plt.cm.ScalarMappable(cmap='Greens',norm=plt.Normalize(vmin=vmin,vmax=vmax␣
 ↪))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_from_2011_Green_Vegetation.plot(column=variable,␣
 ↪cmap='Greens', linewidth=0.8, ax=ax,edgecolor='0.5',
                                          figsize = (150,450))
count= 0
for x, y, label in zip(merged_Bangalore_from_2011_Green_Vegetation.centroid.x,␣
 ↪merged_Bangalore_from_2011_Green_Vegetation.centroid.y,
                       merged_Bangalore_from_2011_Green_Vegetation.WARD_NAME ):
    if label not in lst:
        continue
    count = count + 1
#     print(x,y,label)
    ax1.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 45)
plt.show()
```



The chloropleth above show the mean NDVI mean of the wards of Banglore over the time period of 2011-2020. The wards which show a 15 % decrease in NDVI indices are depicted in the image

[56]:
```
# PLotting 2001-20
lst1 = Green_Vegetation_wards.sort_values(by = ["NDVI_mean"], ascending =␣
 ↪False).head(5).index.tolist()
lst1 = [Bangalore_Wards_List.loc[i][0] for i in lst1]

# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
```

```
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Green_Vegetation from␣
 ↪2000-20',fontdict={'fontsize': '120', 'fontweight' : '3'})

sm = plt.cm.ScalarMappable(cmap='Greens',norm=plt.Normalize(vmin=vmin,vmax=vmax␣
 ↪))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_Green_Vegetation.plot(column=variable, cmap='Greens',␣
 ↪linewidth=0.8, ax=ax,edgecolor='0.5', figsize = (150,450))

count= 0
for x, y, label in zip(merged_Bangalore_Green_Vegetation.centroid.x,␣
 ↪merged_Bangalore_Green_Vegetation.centroid.y,
                       merged_Bangalore_Green_Vegetation.WARD_NAME ):
    if label not in lst1:
        continue
    count = count + 1
    if count % 3 == 0:
#     print(x,y,label)
        ax1.annotate(label, xy=(x, y+0.00001), xytext=(3, 3),␣
 ↪textcoords="offset points",fontsize = 75, color  ="Yellow")
        continue
    ax1.annotate(label, xy=(x, y-0.00001), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 75, color  ="Yellow")
plt.show()
print(lst1)
```




```
['Varthuru', 'Agaram', 'Hemmigepura', 'Kuvempu Nagar', 'Hagadur']
```

The five Wards labeled in the chloropleth have the highest greenery cover according to he NDVI values.

Variable 5: Precipitation - NEX Dataset

Precipitation is rain, snow, sleet, or hail — any kind of weather condition where something's falling from the sky. Precipitation has to do with things falling down, and not just from the sky. It's also what happens in chemical reactions when a solid settles to the bottom of a solution.

[57]:
```
Precipitation =pd.read_csv('Bangalore/NEX_GDDP_precipitation.csv')
Precipitation.date = pd.to_datetime(Precipitation.date)
```

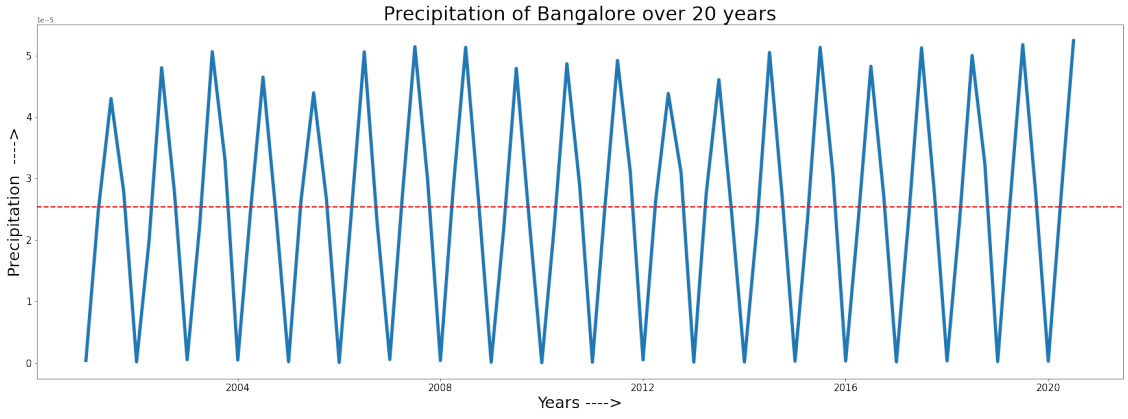
```
Precipitation.rename(columns={'mean':'Precipitation'}, inplace=True)

Precipitation = Precipitation[["WARD_NO","date","Precipitation"]]
Precipitation
```

[57]:

	WARD_NO	date	Precipitation
0	24	2001-01-01	4.900000e-08
1	27	2001-01-01	4.900000e-08
2	28	2001-01-01	4.900000e-08
3	29	2001-01-01	4.900000e-08
4	30	2001-01-01	4.900000e-08
…	…	…	…
15637	172	2020-07-01	5.220000e-05
15638	26	2020-07-01	5.290000e-05
15639	25	2020-07-01	5.290000e-05
15640	86	2020-07-01	5.220000e-05
15641	198	2020-07-01	5.160000e-05

[15642 rows x 3 columns]

[58]:
```
Precipitation_date = Precipitation.groupby("date").mean()
plt.figure(figsize = (30,10))
plt.plot(Precipitation_date.index,Precipitation_date.Precipitation , lw = 5)
plt.title("Precipitation of Bangalore over 20 years", fontsize = 30)
plt.ylabel("Precipitation  ---->", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.axhline(Precipitation_date["Precipitation"].mean(), color = "r", lw = 2,␣
 ↪linestyle = "dashed")
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()
```

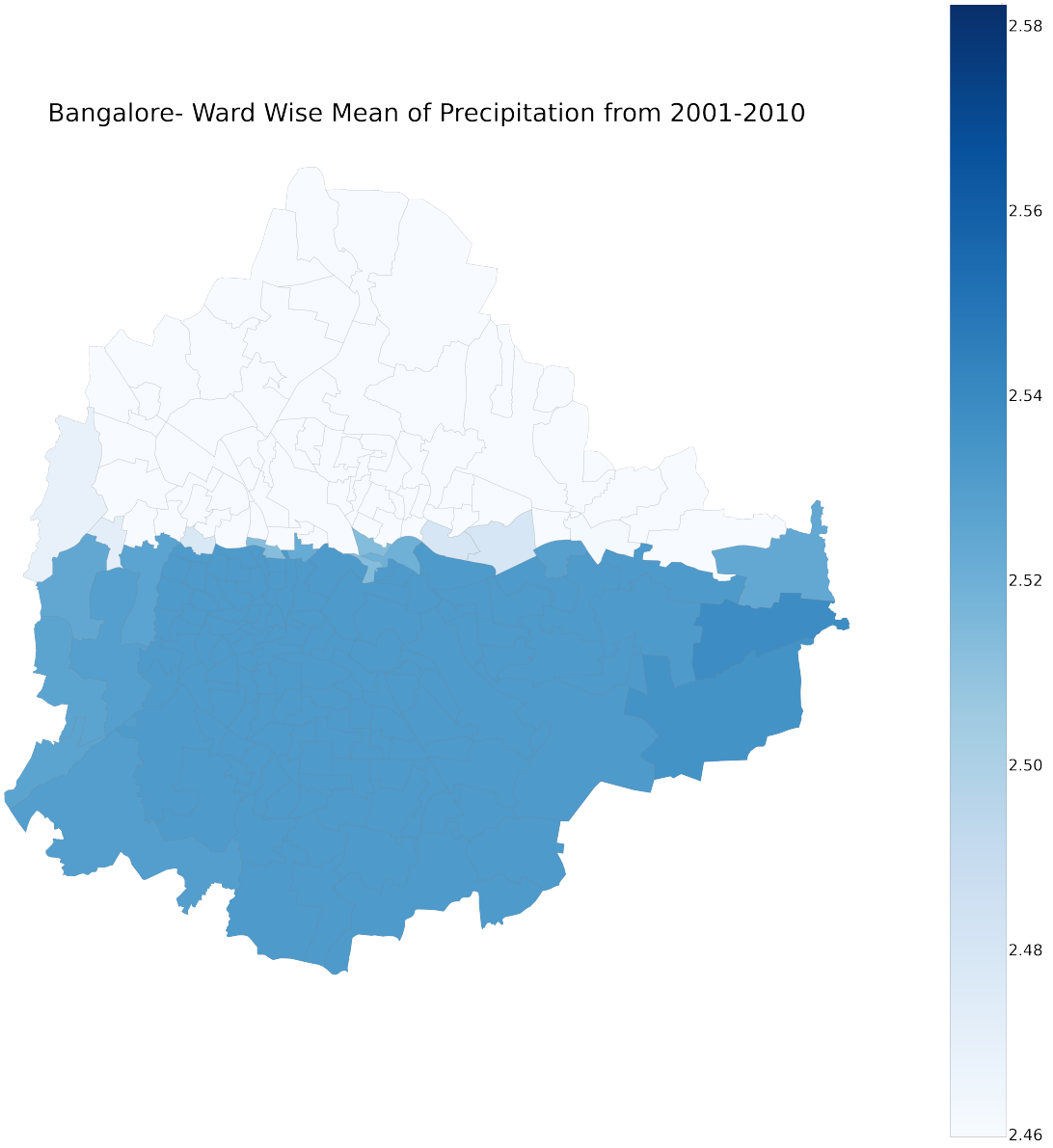


The above graph show the variation of precipiation in Bangalore over the time period of 2001-2020

[59]:
```
Precipitation_till_2010 = Precipitation[(Precipitation["date"]>= "20010101") &␣
↪(Precipitation["date"]< "20110101")]
Precipitation_till_2010 = Precipitation_till_2010.groupby("WARD_NO").mean()
Precipitation_from_2011 = Precipitation[(Precipitation["date"]>= "20110101")]
Precipitation_from_2011 = Precipitation_from_2011.groupby("WARD_NO").mean()
Precipitation_wards = Precipitation.groupby("WARD_NO").mean()
```

[60]:
```
merged_Bangalore_till_2010_Precipitation = map_Bangalore.
↪merge(Precipitation_till_2010, left_on = "WARD_NO",right_index = True)
merged_Bangalore_from_2011_Precipitation = map_Bangalore.
↪merge(Precipitation_from_2011, left_on = "WARD_NO",right_index = True)
merged_Bangalore_Precipitation       = map_Bangalore.merge(Precipitation_wards,␣
↪left_on = "WARD_NO",right_index = True)
```

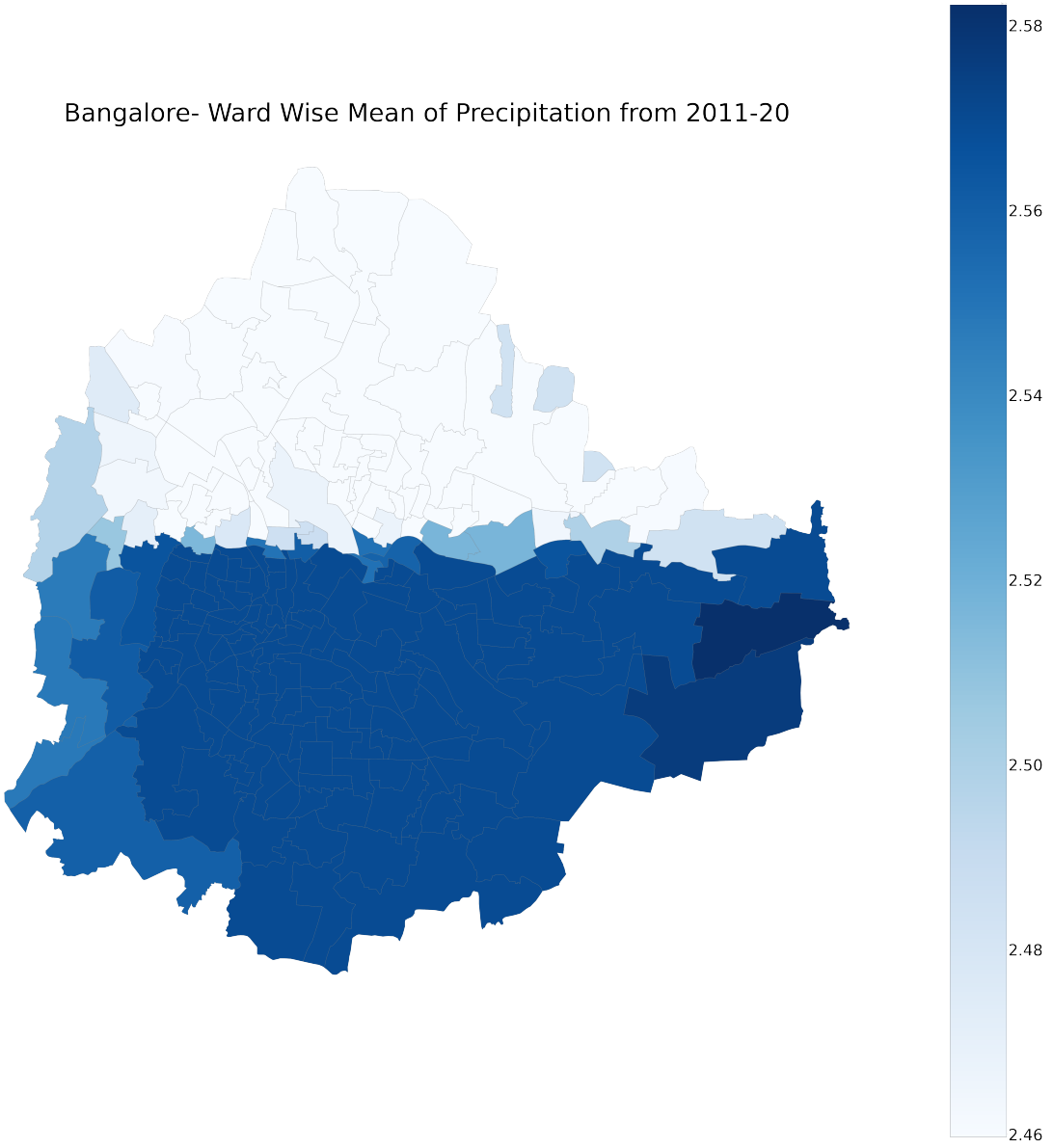
[61]:
```
# Plotting 2001-10

# Precipitation_wards_1 = Precipitation.groupby("WARD_NO").mean()
#␣
↪print(len(Precipitation_till_2010_Precipitation),len(Precipitation_from_2011_Precipitation)
# fig, (ax1,ax2) = plt.subplots(1,2,figsize = (8,8), sharex = True, sharey=True)
variable = 'Precipitation'
# # set the range for the choropleth values
vmin, vmax = merged_Bangalore_Precipitation["Precipitation"].min() ,␣
↪merged_Bangalore_Precipitation["Precipitation"].max()
# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Precipitation from 2001-2010',
             fontdict={'fontsize': '120', 'fontweight' : '3'})


sm = plt.cm.ScalarMappable(cmap='Blues',norm=plt.Normalize(vmin=vmin
                                                           ,vmax=vmax ))
# empty array for the data range
sm.set_array([])

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
merged_Bangalore_till_2010_Precipitation.plot(column=variable,␣
↪cmap='Blues',vmin = vmin, vmax = vmax,linewidth=0.8, ax=ax,
                                edgecolor='0.5', figsize = (150,450))
plt.show()
```



The chloreopleth shows the mean values of precipiation of all wards in Banglore from 2001 till 2010

[62]:
```
# Plotting 2011-20
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Precipitation from 2011-20'
             ,fontdict={'fontsize': '120', 'fontweight' : '3'})
```

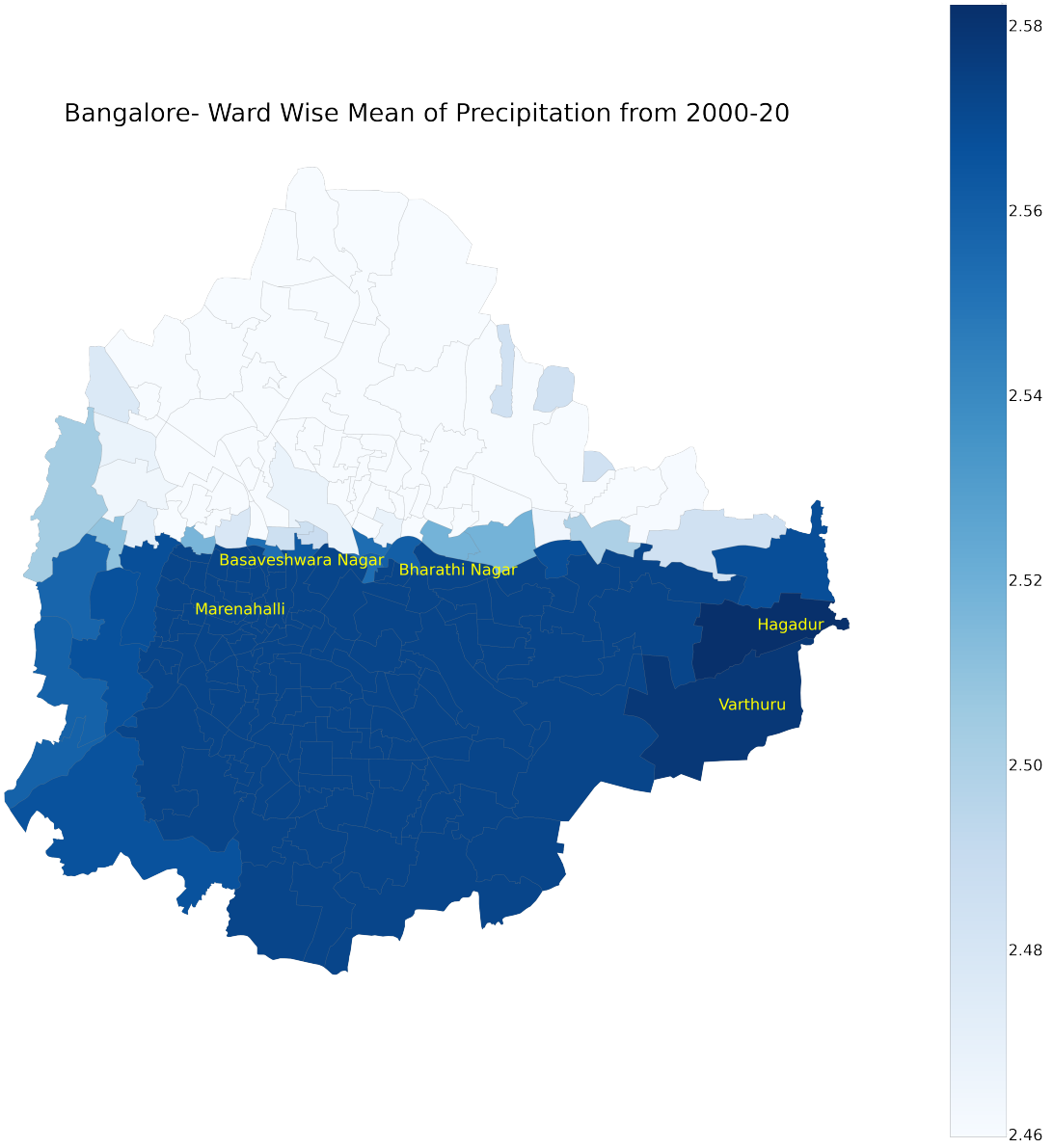
```
sm = plt.cm.ScalarMappable(cmap='Blues',norm=plt.Normalize(vmin=vmin,vmax=vmax␣
 ↪))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_from_2011_Precipitation.plot(column=variable,␣
 ↪cmap='Blues', linewidth=0.8, ax=ax,edgecolor='0.5',
                                          figsize = (150,450))
count= 0
# for x, y, label in zip(merged_Bangalore_from_2011_Precipitation.centroid.x,␣
 ↪merged_Bangalore_from_2011_Precipitation.centroid.y,
#                         merged_Bangalore_from_2011_Precipitation.WARD_NAME ):
#     if label not in lst:
#         continue
#     count = count + 1
# #     print(x,y,label)
#     ax1.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 75)
plt.show()
```



The chloreopleth shows the mean values of precipiation of all wards in Banglore from 2011 till 2020

[63]:
```
# Plotting 2001-20
lst1 = Precipitation_wards.sort_values(by = ["Precipitation"], ascending =␣
 ↪False).head(5).index.tolist()
lst1 = [Bangalore_Wards_List.loc[i][0] for i in lst1]

# create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(100, 100))
# remove the axis
ax.axis('off')
```

```
# add a title and annotation
ax.set_title('Bangalore- Ward Wise Mean of Precipitation from␣
 ↪2000-20',fontdict={'fontsize': '120', 'fontweight' : '3'})

sm = plt.cm.ScalarMappable(cmap='Blues',norm=plt.Normalize(vmin=vmin,vmax=vmax␣
 ↪))
# empty array for the data range
sm.set_array([]) # or alternatively sm._A = []. Not sure why this step␣
 ↪is␣,necessary, but many recommends it

bx = fig.colorbar(sm)
bx.ax.tick_params(labelsize=75)
# create map
ax1 = merged_Bangalore_Precipitation.plot(column=variable, cmap='Blues',␣
 ↪linewidth=0.8, ax=ax,edgecolor='0.5', figsize = (150,450))

count= 0
for x, y, label in zip(merged_Bangalore_Precipitation.centroid.x,␣
 ↪merged_Bangalore_Precipitation.centroid.y,
                       merged_Bangalore_Precipitation.WARD_NAME ):
    if label not in lst1:
        continue
    count = count + 1
    if count % 3 == 0:
#     print(x,y,label)
        ax1.annotate(label, xy=(x, y+0.00001), xytext=(3, 3),␣
 ↪textcoords="offset points",fontsize = 75, color = "Yellow")
        continue
    ax1.annotate(label, xy=(x, y-0.00001), xytext=(3, 3), textcoords="offset␣
 ↪points",fontsize = 75,  color = "Yellow")
plt.show()
print("The Five Wards with Maximum Precipitation are : ", lst1)
```




```
The Five Wards with Maximum Precipitation are :  ['Hagadur', 'Varthuru',
'Basaveshwara Nagar', 'Marenahalli', 'Bharathi Nagar']
```

The chloropleth shows the 5 wards with maximum precipitaion in the last 20 years. In can be observed that the precipitaion in higher in the northern half of Bangalore which is similar to the the Temperature. Therefore we can conclude that the presence of Nandhi Hills in th Northern part helps in increased precipitaion and thereby lower temperature.

4 The MASTER Merged Dataset

4.1 Merging Individual Dataframes for Bangalore Wards

[64]:
```
new_d1 = Green_Vegetation.merge(ERA, how = "inner", left_on =␣
 ↪["WARD_NO","date"], right_on = ["WARD_NO","date"] )
new_d1 = new_d1.merge(Precipitation,left_on = ["WARD_NO","date"],right_on =␣
 ↪["WARD_NO","date"])
new_d1 = new_d1.merge(AOD, how = "inner", left_on = ["WARD_NO","date"], ␣
 ↪right_on = ["WARD_NO","date"] )
new_d1.AOD.dropna(inplace = True)
new_d1 = new_d1.merge(EvapoTranspiration, how = "inner", left_on =␣
 ↪["WARD_NO","date"],  right_on = ["WARD_NO","date"] )
new_d1["EvapoTranspiration"].dropna(inplace = True)
# print(new_d1.columns)
new_d1 =␣
 ↪new_d1[["date","WARD_NO","AOD","EvapoTranspiration","Precipitation","EVI_mean","NDVI_mean",
            ␣
 ↪"minimum_2m_air_temperature","Temperature_Mean","maximum_2m_air_temperature"]]␣
 ↪#

new_d1
```

[64]:

	date	WARD_NO	AOD	EvapoTranspiration	Precipitation	\
0	2001-01-01	2	490.087543	61.913493	4.900000e-08	
1	2001-01-01	3	509.689948	59.552162	4.900000e-08	
2	2001-01-01	4	469.909609	64.469620	4.900000e-08	
3	2001-01-01	53	541.013335	55.868946	5.160000e-08	
4	2001-01-01	54	533.113471	60.452495	1.780000e-07	
…	…	…	…	…	…	
6998	2020-01-01	177	454.230848	47.778448	3.530000e-07	
6999	2020-01-01	26	473.860709	57.368084	1.280000e-07	
7000	2020-01-01	25	480.498541	60.905588	1.280000e-07	
7001	2020-01-01	86	468.443644	61.838838	3.530000e-07	
7002	2020-01-01	198	426.522781	70.114378	3.750000e-07	

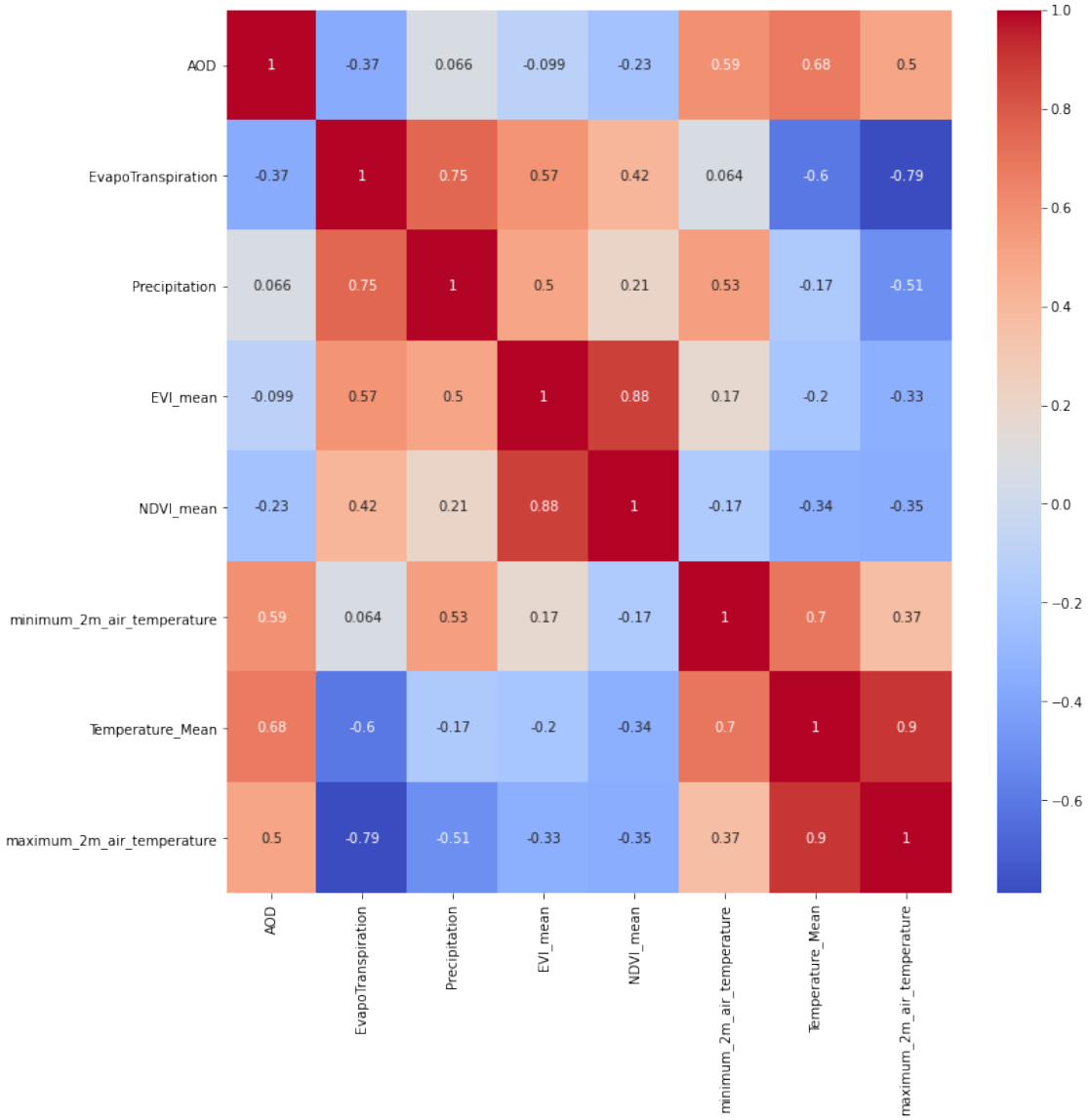
	EVI_mean	NDVI_mean	minimum_2m_air_temperature	Temperature_Mean	\
0	2014.706828	3493.084593	287.801910	297.031036	
1	1903.781523	3179.505877	287.801910	297.031036	
2	2064.841951	3465.578946	287.801910	297.031036	
3	1841.154779	3190.306837	287.803548	297.031351	
4	1992.591517	3431.221221	287.849520	297.020652	
…	…	…	…	…	
6998	1351.049615	2536.052886	290.525635	297.566040	
6999	1597.486161	2816.833160	290.268280	297.484283	
7000	1540.020164	2753.535720	290.268280	297.484283	
7001	1610.859884	2862.603246	290.525635	297.566040	

```
7002  1928.336195  3635.664204                  290.687932        297.702403

      maximum_2m_air_temperature
0                     305.320099
1                     305.320099
2                     305.320099
3                     305.320161
4                     305.307567
…                            …
6998                  305.183746
6999                  305.272675
7000                  305.272675
7001                  305.183746
7002                  305.604715

[7003 rows x 10 columns]
```

4.2 Correlation and MultiVariate Analysis

[65]:
```
plt.figure(figsize= (12,12))
sns.heatmap(new_d1.iloc[:,2:].corr(),annot =True,cmap = "coolwarm")
```

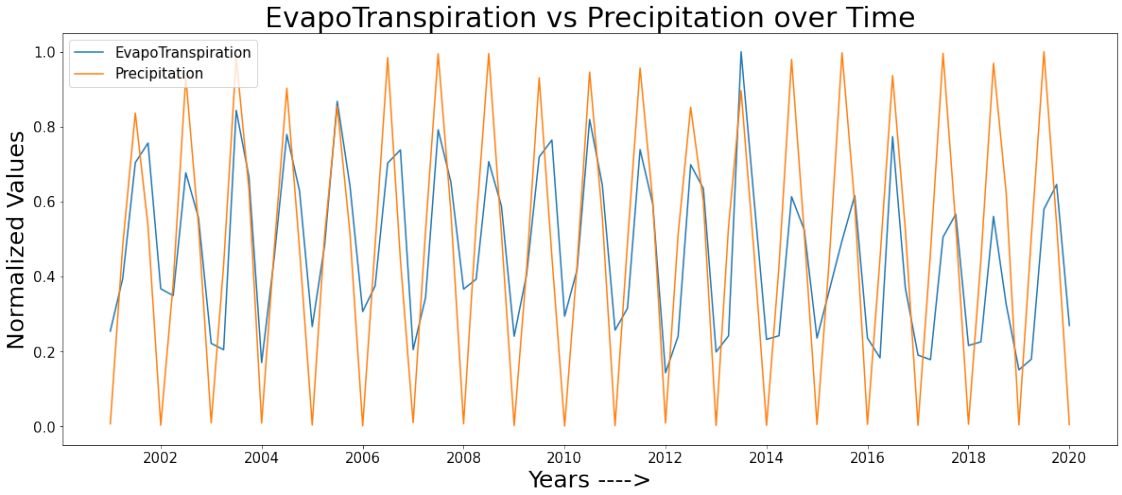
[65]:
```
<AxesSubplot:>
```



Viewing Correlations

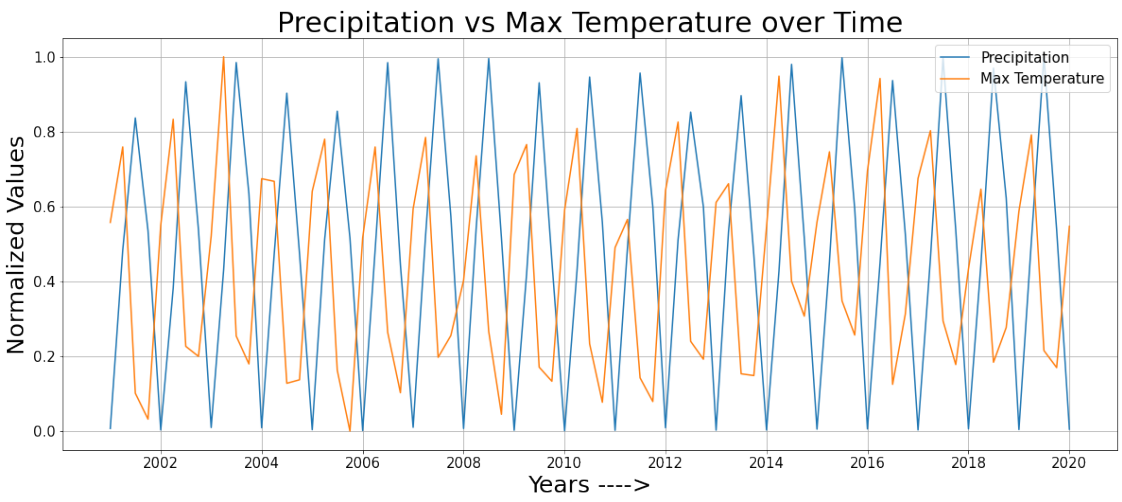
[66]:
```
new_d1_group = new_d1.groupby("date").mean()
plt.figure(figsize = (20,8))
plt.title("EvapoTranspiration vs Precipitation over Time", fontsize = 30)
plt.plot(new_d1_group.index,new_d1_group["EvapoTranspiration"]/
 ↪max(new_d1_group["EvapoTranspiration"]), label = "EvapoTranspiration")
plt.plot(new_d1_group.index,new_d1_group["Precipitation"]/
 ↪max(new_d1_group["Precipitation"]), label = "Precipitation")
plt.ylabel("Normalized Values", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.legend(fontsize = 15)
```

```
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
# plt.grid()
plt.show()
```



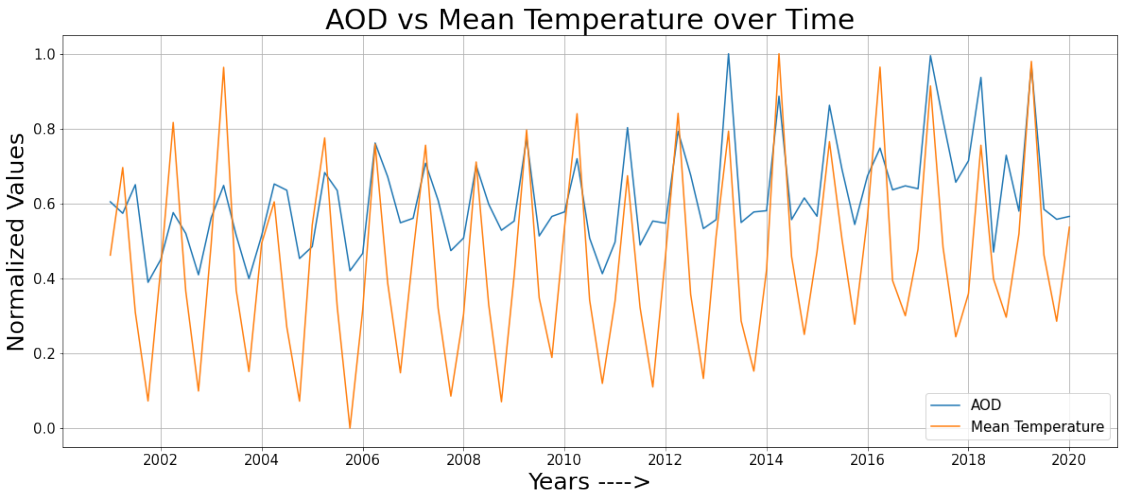
EvapoTranspiration enjoys a strong Direct Correlation with Precipitation. And we can see more they peak around the same period every year.

[67]:
```
new_d1_group = new_d1.groupby("date").mean()
plt.figure(figsize = (20,8))
plt.plot(new_d1_group.index,new_d1_group["Precipitation"]/
 ↪max(new_d1_group["Precipitation"]), label = "Precipitation",)
plt.plot(new_d1_group.
 ↪index,(new_d1_group["maximum_2m_air_temperature"]-min(new_d1_group["maximum_2m_air_temperat
 ↪(max(new_d1_group["maximum_2m_air_temperature"])-␣
 ↪min(new_d1_group["maximum_2m_air_temperature"])), label = "Max Temperature")
plt.title("Precipitation vs Max Temperature over Time", fontsize = 30)
plt.ylabel("Normalized Values", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.legend(fontsize = 15)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.grid()
```



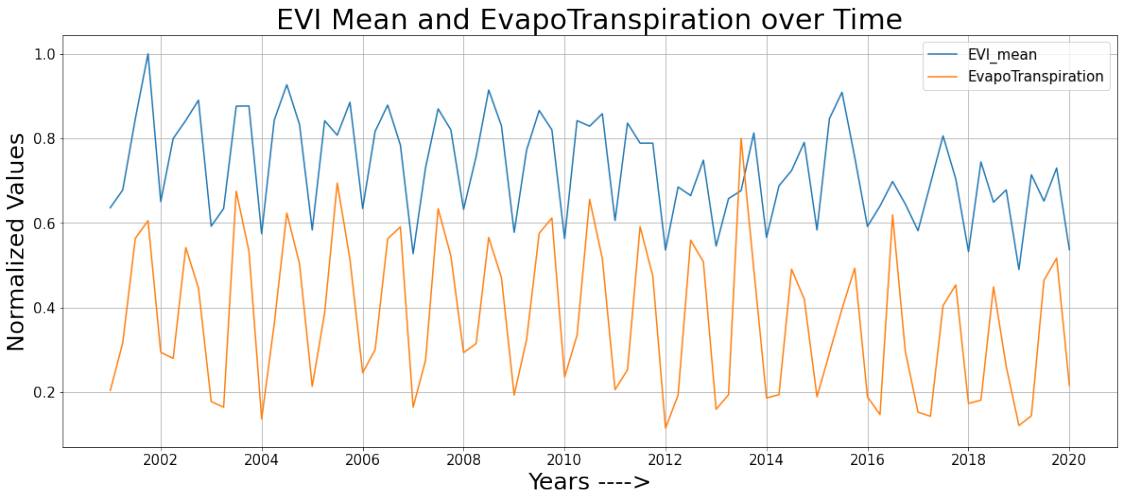
We can observe that Precipitation results in lowering of the temperatures. Their peaks are separated by a time frame every year.

[68]:
```
new_d1_group = new_d1.groupby("date").mean()
plt.figure(figsize = (20,8))
plt.plot(new_d1_group.index,new_d1_group["AOD"]/max(new_d1_group["AOD"]), label␣
↪= "AOD")
plt.plot(new_d1_group.
↪index,(new_d1_group["Temperature_Mean"]-min(new_d1_group["Temperature_Mean"]))/
↪(max(new_d1_group["Temperature_Mean"])-␣
↪min(new_d1_group["Temperature_Mean"])), label = "Mean Temperature")
plt.title("AOD vs Mean Temperature over Time", fontsize = 30)
plt.ylabel("Normalized Values", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.legend(fontsize = 15)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.grid()
```



We can observe from the graphs that, whenever AOD and Mean Temperature peak around the same time, and AOD has an impact on Mean Temperature.

[69]:
```
new_d1_group = new_d1.groupby("date").mean()
plt.figure(figsize = (20,8))
plt.plot(new_d1_group.index,new_d1_group["EVI_mean"]/
 ↪max(new_d1_group["EVI_mean"]), label = "EVI_mean")
plt.plot(new_d1_group.index,new_d1_group["EvapoTranspiration"]/(1.
 ↪25*max(new_d1_group["EvapoTranspiration"])), label = "EvapoTranspiration")
plt.grid()
plt.title("EVI Mean and EvapoTranspiration over Time", fontsize = 30)
plt.ylabel("Normalized Values", fontsize = 25)
plt.xlabel("Years ---->", fontsize = 25)
plt.legend(fontsize = 15)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
plt.show()
```



EVI Mean is an indicator of Vegetation Cover of a Region. They peak around the same time every year. But there are other factors affecting EvapoTranspiration too, like availability of Water Bodies, and this is the reason for the bumpy graph. But still we can observe that, more the vegetation cover, more the evapotranspiration.

5 Regression Analysis

Regression analysis is a set of statistical methods used for the estimation of relationships between a dependent variable and one or more independent variables. It can be utilized to assess the strength of the relationship between variables and for modeling the future relationship between them.

Here, we are planning to estimate Mean Temperature of Bangalore from 2020-23 using the other input variables AOD, EvapoTranspiration, NDVI_mean, EVI_mean, Precipitation. We will avoid feeding Min Temperature and Max Temperature as this could bias the Model.

5.1 Feature Selection

[70]:
```
new_d1.var()
```

[70]:
```
WARD_NO                        4.324660e+03
AOD                            1.482869e+04
EvapoTranspiration             2.986945e+03
Precipitation                  3.003762e-10
EVI_mean                       2.473054e+05
NDVI_mean                      4.997229e+05
minimum_2m_air_temperature     5.954714e+00
Temperature_Mean               2.381576e+00
maximum_2m_air_temperature     4.697102e+00
dtype: float64
```

Here we remove Precipitation as it is Quasi Static and also EVI_Mean (because anyways its directly related to NDVI Mean) from our input variables.

We are left with **three Input Variables** - AOD - EvapoTranspiration - NDVI_mean

Output Variable - Mean Temperature

[71]:
```
# We remove Precipitation since it has quasi-static
new_d1_group1 = new_d1.groupby("date").mean()
new_d1_group1["Month"] = new_d1_group1.index.month
new_d1_group1["Year"] = new_d1_group1.index.year
new_d1_group1 = new_d1_group1[["Month","Year","AOD",␣
 ↪"EvapoTranspiration",'NDVI_mean', "Temperature_Mean"]]
new_d1_group = new_d1_group1.iloc[:76,:]
new_d1_group
```

[71]:

date	Month	Year	AOD	EvapoTranspiration	NDVI_mean	Temperature_Mean
2001-01-01	1	2001	497.733339	61.383345	3217.929364	297.089799
2001-04-01	4	2001	472.429076	95.357357	2850.632338	298.533487
2001-07-01	7	2001	535.345818	169.862813	3105.716919	296.147626
2001-10-01	10	2001	320.603361	182.502444	4672.716053	294.687213
2002-01-01	1	2002	370.794589	88.626164	3316.753783	296.879465
…	…	…	…	…	…	…
2018-10-01	10	2018	600.081920	78.565652	3555.548772	296.066873
2019-01-01	1	2019	477.160752	36.395282	2768.237223	297.439473
2019-04-01	4	2019	798.402262	43.330479	3216.676352	300.282479
2019-07-01	7	2019	481.168087	139.953868	2905.995299	297.091113
2019-10-01	10	2019	459.231755	155.813150	3815.077806	296.000358

[76 rows x 6 columns]

[72]:
```
# Summary Statistics of Train Data
new_d1_group.describe().iloc[:,2:]
```

[72]:

	AOD	EvapoTranspiration	NDVI_mean	Temperature_Mean
count	76.000000	76.000000	76.000000	76.000000
mean	506.677663	113.177366	3352.562857	297.004762
std	113.487330	53.709503	467.291230	1.550323
min	320.603361	34.540702	2540.653055	294.239685
25%	433.415709	58.371416	2980.722467	296.051691
50%	476.398577	104.758563	3322.053241	296.726067
75%	557.788886	156.217815	3664.481829	297.775688
max	823.351477	241.291722	4672.716053	300.408515

5.2 Standardizing Input for the curves

Our Plan is to Predict the Future AOD, EvapoTranspiration and NDVI_Mean.

And with those inputs, predict Future Mean Temperature.

[73]:
```
# Future Predictions Dataframe
future_predictions = pd.DataFrame(index =␣
↪["2020-01-01","2020-04-01","2020-07-01","2020-10-01",
                           "2021-01-01","2021-04-01","2021-07-01","2021-10-01",
                           "2022-01-01","2022-04-01","2022-07-01","2022-10-01",
                          ␣
↪"2023-01-01","2023-04-01","2023-07-01","2023-10-01"],
                    columns =␣
↪["AOD","EvapoTranspiration","NDVI_mean","Temperature_Mean"])
future_predictions.index = pd.to_datetime(future_predictions.index)
future_predictions["Month"] = future_predictions.index.month
future_predictions["Year"] = future_predictions.index.year
```

[74]:
```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
X =  new_d1_group[["Month","Year"]]

scaler =  StandardScaler()
X_i = scaler.fit_transform(X)
X_test = scaler.transform(future_predictions[["Month","Year"]])
```

5.3 Regression Analysis for Future Evapotranspiration

[75]:
```
# Evapotranspiration
clf_evapo_0 = LinearRegression()
clf_evapo_0.fit(X_i[::4], (new_d1_group["EvapoTranspiration"].to_numpy())[::4])

clf_evapo_1 = LinearRegression()
clf_evapo_1.fit(X_i[1::4], (new_d1_group["EvapoTranspiration"].to_numpy())[1::
↪4])
```

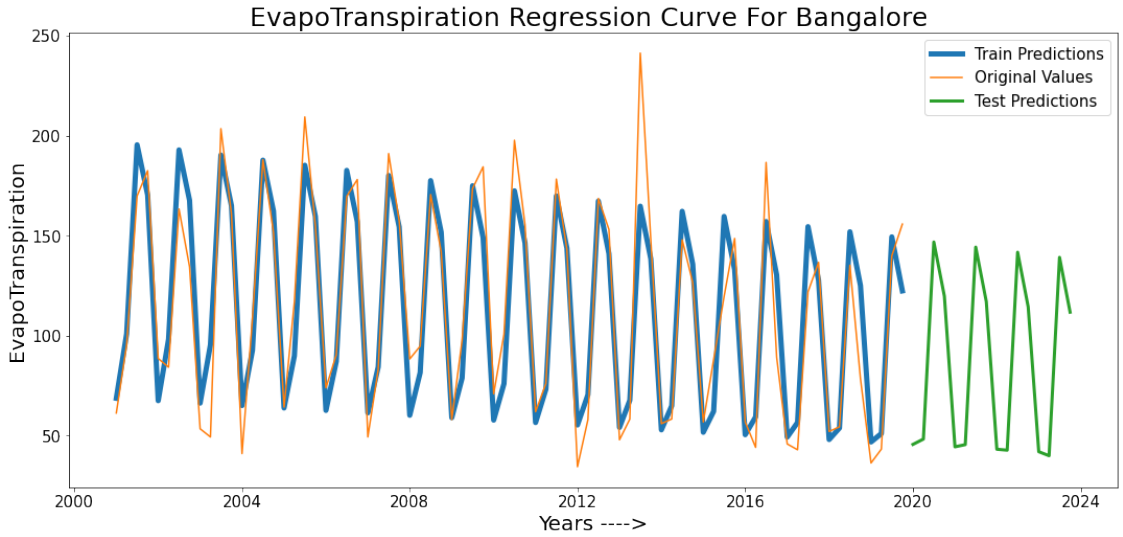
```
clf_evapo_2 = LinearRegression()
clf_evapo_2.fit(X_i[2::4], (new_d1_group["EvapoTranspiration"].to_numpy())[2::
↪4])

clf_evapo_3 = LinearRegression()
clf_evapo_3.fit(X_i[3::4], (new_d1_group["EvapoTranspiration"].to_numpy())[3::
↪4])

evapo_tp_0 = clf_evapo_0.predict(X_i[::4])
evapo_tp_1 = clf_evapo_1.predict(X_i[1::4])
evapo_tp_2 = clf_evapo_2.predict(X_i[2::4])
evapo_tp_3 = clf_evapo_3.predict(X_i[3::4])

evapo_test_0 = clf_evapo_0.predict(X_test[::4])
evapo_test_1 = clf_evapo_1.predict(X_test[1::4])
evapo_test_2 = clf_evapo_2.predict(X_test[2::4])
evapo_test_3 = clf_evapo_3.predict(X_test[3::4])

evapo_train_predictions = []
evapo_test_predictions  = []

for i in range(19):
    evapo_train_predictions.append(evapo_tp_0[i])
    evapo_train_predictions.append(evapo_tp_1[i])
    evapo_train_predictions.append(evapo_tp_2[i])
    evapo_train_predictions.append(evapo_tp_3[i])

for i in range(4):
    evapo_test_predictions.append(evapo_test_0[i])
    evapo_test_predictions.append(evapo_test_1[i])
    evapo_test_predictions.append(evapo_test_2[i])
    evapo_test_predictions.append(evapo_test_3[i])

    # clf.predict([["07.0","2021.0"]])

evapo_r2_score =␣
↪r2_score(new_d1_group["EvapoTranspiration"],evapo_train_predictions)
print("The R2 Score for our model is ",evapo_r2_score)

plt.figure(figsize = (18,8))
plt.plot(new_d1_group.index,evapo_train_predictions, lw = 5, label = "Train␣
↪Predictions")
plt.plot(new_d1_group.index,new_d1_group["EvapoTranspiration"], label =␣
↪"Original Values")
plt.title("EvapoTranspiration Regression Curve For Bangalore ", fontsize = 25)
```

```
plt.plot(future_predictions.index, evapo_test_predictions , label = "Test␣
↪Predictions", lw = 3)
plt.ylabel("EvapoTranspiration", fontsize = 20)
plt.xlabel("Years ---->", fontsize = 20)
plt.legend(fontsize = 15)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
future_predictions["EvapoTranspiration"] = evapo_test_predictions
future_predictions
```

```
The R2 Score for our model is  0.8551674278721275
```

[75]:

	AOD	EvapoTranspiration	NDVI_mean	Temperature_Mean	Month	Year
2020-01-01	NaN	45.685520	NaN	NaN	1	2020
2020-04-01	NaN	48.346896	NaN	NaN	4	2020
2020-07-01	NaN	146.862655	NaN	NaN	7	2020
2020-10-01	NaN	119.759561	NaN	NaN	10	2020
2021-01-01	NaN	44.471675	NaN	NaN	1	2021
2021-04-01	NaN	45.574263	NaN	NaN	4	2021
2021-07-01	NaN	144.305758	NaN	NaN	7	2021
2021-10-01	NaN	117.097452	NaN	NaN	10	2021
2022-01-01	NaN	43.257831	NaN	NaN	1	2022
2022-04-01	NaN	42.801630	NaN	NaN	4	2022
2022-07-01	NaN	141.748861	NaN	NaN	7	2022
2022-10-01	NaN	114.435343	NaN	NaN	10	2022
2023-01-01	NaN	42.043986	NaN	NaN	1	2023
2023-04-01	NaN	40.028998	NaN	NaN	4	2023
2023-07-01	NaN	139.191964	NaN	NaN	7	2023
2023-10-01	NaN	111.773234	NaN	NaN	10	2023



We can see that our model reasonably fits the Original Values Curve and we can observe the decreasing Trend of EvapoTranspiration which could be indicating Higher Temperatures and Lesser Vegetation Cover as some of the reasons.

5.4 Regression Analysis for Future NDVI Mean

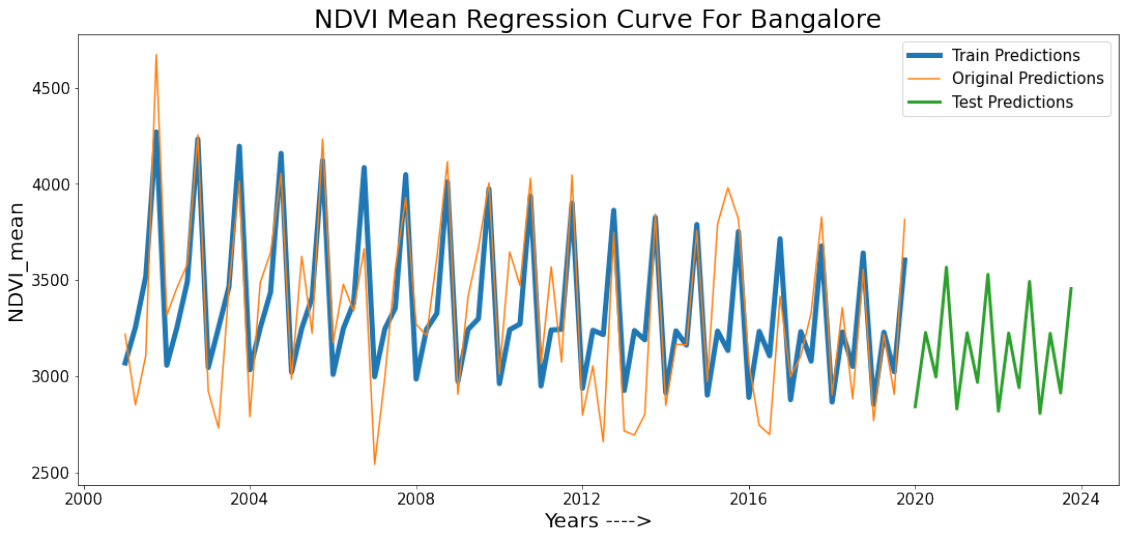
[76]:
```
# NDVI_mean
plt.figure(figsize = (18,8))

clf_ndvi_0 = LinearRegression()
clf_ndvi_0.fit(X_i[::4], (new_d1_group["NDVI_mean"].to_numpy())[::4])

clf_ndvi_1 = LinearRegression()
clf_ndvi_1.fit(X_i[1::4], (new_d1_group["NDVI_mean"].to_numpy())[1::4])

clf_ndvi_2 = LinearRegression()
clf_ndvi_2.fit(X_i[2::4], (new_d1_group["NDVI_mean"].to_numpy())[2::4])

clf_ndvi_3 = LinearRegression()
clf_ndvi_3.fit(X_i[3::4], (new_d1_group["NDVI_mean"].to_numpy())[3::4])

ndvi_tp_0 = clf_ndvi_0.predict(X_i[::4])
ndvi_tp_1 = clf_ndvi_1.predict(X_i[1::4])
ndvi_tp_2 = clf_ndvi_2.predict(X_i[2::4])
ndvi_tp_3 = clf_ndvi_3.predict(X_i[3::4])

ndvi_test_0 = clf_ndvi_0.predict(X_test[::4])
ndvi_test_1 = clf_ndvi_1.predict(X_test[1::4])
ndvi_test_2 = clf_ndvi_2.predict(X_test[2::4])
ndvi_test_3 = clf_ndvi_3.predict(X_test[3::4])

ndvi_train_predictions = []
ndvi_test_predictions  = []

for i in range(19):
    ndvi_train_predictions.append(ndvi_tp_0[i])
    ndvi_train_predictions.append(ndvi_tp_1[i])
    ndvi_train_predictions.append(ndvi_tp_2[i])
    ndvi_train_predictions.append(ndvi_tp_3[i])

for i in range(4):
    ndvi_test_predictions.append(ndvi_test_0[i])
    ndvi_test_predictions.append(ndvi_test_1[i])
    ndvi_test_predictions.append(ndvi_test_2[i])
    ndvi_test_predictions.append(ndvi_test_3[i])

ndvi_r2_score = r2_score(new_d1_group["NDVI_mean"],ndvi_train_predictions)
```

```
print("The R2 score for NDVI Mean = ",ndvi_r2_score)

    # clf.predict([["07.0","2021.0"]])
plt.plot(new_d1_group.index,ndvi_train_predictions, lw = 5, label = "Train␣
 ↪Predictions")
plt.plot(new_d1_group.index,new_d1_group["NDVI_mean"], label = "Original␣
 ↪Predictions")
plt.title("NDVI Mean Regression Curve For Bangalore", fontsize = 25)
plt.ylabel("NDVI_mean", fontsize = 20)
plt.xlabel("Years ---->", fontsize = 20)
plt.plot(future_predictions.index, ndvi_test_predictions , label = "Test␣
 ↪Predictions", lw = 3)
plt.legend(fontsize = 15)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)

future_predictions["NDVI_mean"] = ndvi_test_predictions
future_predictions
```

```
The R2 score for NDVI Mean =  0.6754288842109899
```

[76]:

	AOD	EvapoTranspiration	NDVI_mean	Temperature_Mean	Month	Year
2020-01-01	NaN	45.685520	2841.242598	NaN	1	2020
2020-04-01	NaN	48.346896	3226.741456	NaN	4	2020
2020-07-01	NaN	146.862655	2995.178019	NaN	7	2020
2020-10-01	NaN	119.759561	3566.398475	NaN	10	2020
2021-01-01	NaN	44.471675	2829.284085	NaN	1	2021
2021-04-01	NaN	45.574263	3225.278003	NaN	4	2021
2021-07-01	NaN	144.305758	2967.574291	NaN	7	2021
2021-10-01	NaN	117.097452	3529.355081	NaN	10	2021
2022-01-01	NaN	43.257831	2817.325571	NaN	1	2022
2022-04-01	NaN	42.801630	3223.814551	NaN	4	2022
2022-07-01	NaN	141.748861	2939.970563	NaN	7	2022
2022-10-01	NaN	114.435343	3492.311687	NaN	10	2022
2023-01-01	NaN	42.043986	2805.367058	NaN	1	2023
2023-04-01	NaN	40.028998	3222.351098	NaN	4	2023
2023-07-01	NaN	139.191964	2912.366835	NaN	7	2023
2023-10-01	NaN	111.773234	3455.268294	NaN	10	2023



The regression curve fits reasonably well once again. We can observe the vegetation cover is decreasing slowly and steadily with a few spikes.

5.5 Regression Analysis for Future AOD

[77]:
```
# AOD
plt.figure(figsize = (20,10))
clf_aod_0 = LinearRegression()
clf_aod_0.fit(X_i[::4], (new_d1_group["AOD"].to_numpy())[::4])


clf_aod_1 = LinearRegression()
clf_aod_1.fit(X_i[1::4], (new_d1_group["AOD"].to_numpy())[1::4])

clf_aod_2 = LinearRegression()
clf_aod_2.fit(X_i[2::4], (new_d1_group["AOD"].to_numpy())[2::4])

clf_aod_3 = LinearRegression()
clf_aod_3.fit(X_i[3::4], (new_d1_group["AOD"].to_numpy())[3::4])

aod_tp_0 = clf_aod_0.predict(X_i[::4])
aod_tp_1 = clf_aod_1.predict(X_i[1::4])
aod_tp_2 = clf_aod_2.predict(X_i[2::4])
aod_tp_3 = clf_aod_3.predict(X_i[3::4])

aod_test_0 = clf_aod_0.predict(X_test[::4])
aod_test_1 = clf_aod_1.predict(X_test[1::4])
aod_test_2 = clf_aod_2.predict(X_test[2::4])
aod_test_3 = clf_aod_3.predict(X_test[3::4])
```

```
aod_train_predictions = []
aod_test_predictions  = []

for i in range(19):
    aod_train_predictions.append(aod_tp_0[i])
    aod_train_predictions.append(aod_tp_1[i])
    aod_train_predictions.append(aod_tp_2[i])
    aod_train_predictions.append(aod_tp_3[i])

for i in range(4):
    aod_test_predictions.append(aod_test_0[i])
    aod_test_predictions.append(aod_test_1[i])
    aod_test_predictions.append(aod_test_2[i])
    aod_test_predictions.append(aod_test_3[i])

    # clf.predict([["07.0","2021.0"]])
aod_r2_score = r2_score(new_d1_group["AOD"],aod_train_predictions)
print("The R2 Score for this AOD Model = ",aod_r2_score)

plt.plot(new_d1_group.index,aod_train_predictions, lw = 5, label = "Train␣
 ↪Predictions")
plt.plot(new_d1_group.index,new_d1_group["AOD"], label = "Original Values")
plt.plot(future_predictions.index, aod_test_predictions, label = "Future␣
 ↪Predictions", lw = 3)
plt.title("AOD Regression Curve For Bangalore", fontsize = 40)
plt.ylabel("AOD", fontsize = 20)
plt.xlabel("Years ---->", fontsize = 20)
plt.legend(fontsize = 15)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)

future_predictions["AOD"] = aod_test_predictions
future_predictions
```

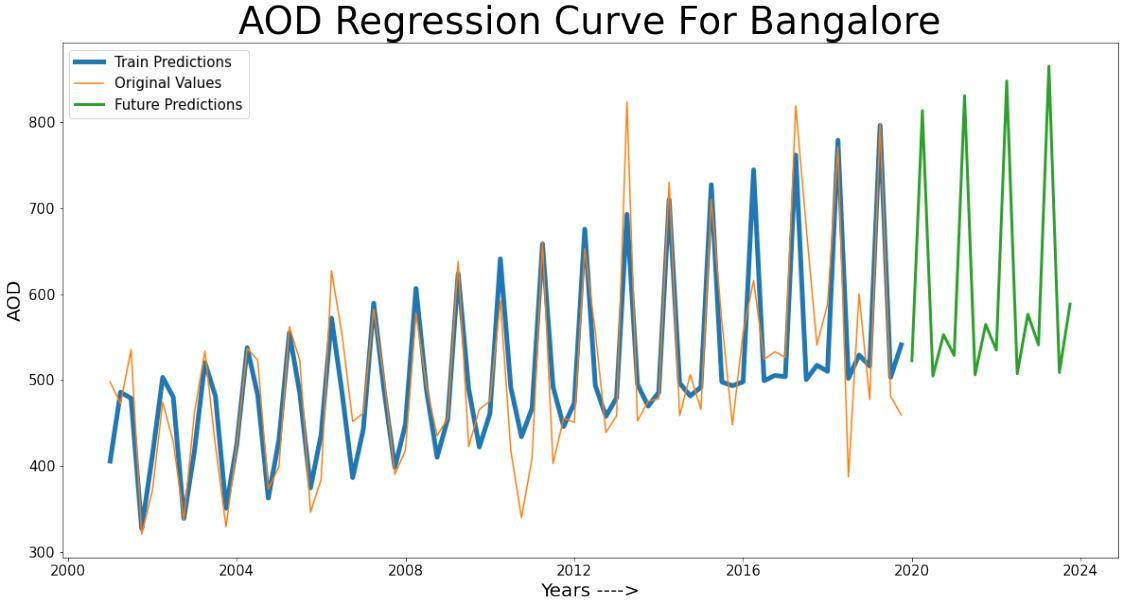
```
The R2 Score for this AOD Model =  0.7872749465378956
```

[77]:

	AOD	EvapoTranspiration	NDVI_mean	Temperature_Mean	\
2020-01-01	522.355691	45.685520	2841.242598	NaN	
2020-04-01	813.442515	48.346896	3226.741456	NaN	
2020-07-01	504.522141	146.862655	2995.178019	NaN	
2020-10-01	552.757363	119.759561	3566.398475	NaN	
2021-01-01	528.501469	44.471675	2829.284085	NaN	
2021-04-01	830.688162	45.574263	3225.278003	NaN	
2021-07-01	505.883661	144.305758	2967.574291	NaN	
2021-10-01	564.641124	117.097452	3529.355081	NaN	
2022-01-01	534.647247	43.257831	2817.325571	NaN	

2022-04-01	847.933808	42.801630	3223.814551	NaN
2022-07-01	507.245181	141.748861	2939.970563	NaN
2022-10-01	576.524885	114.435343	3492.311687	NaN
2023-01-01	540.793025	42.043986	2805.367058	NaN
2023-04-01	865.179455	40.028998	3222.351098	NaN
2023-07-01	508.606701	139.191964	2912.366835	NaN
2023-10-01	588.408646	111.773234	3455.268294	NaN

	Month	Year
2020-01-01	1	2020
2020-04-01	4	2020
2020-07-01	7	2020
2020-10-01	10	2020
2021-01-01	1	2021
2021-04-01	4	2021
2021-07-01	7	2021
2021-10-01	10	2021
2022-01-01	1	2022
2022-04-01	4	2022
2022-07-01	7	2022
2022-10-01	10	2022
2023-01-01	1	2023
2023-04-01	4	2023
2023-07-01	7	2023
2023-10-01	10	2023



The Regression Curve fits reasonably well and we can see that AOD has been increasing over the

past few years and will continue to increase.

[78]:
```
future_predictions = future_predictions[["Month","Year","AOD",␣
↪"EvapoTranspiration", 'NDVI_mean', "Temperature_Mean"]]
# future_predictions = future_predictions[["Month","Year",'NDVI_mean',␣
↪"Temperature_Mean"]]
```

Now, we use the future AOD, NDVI_Mean and EvapoTranspiration to Predict Mean Temperature

5.6 Regression Analysis for Predicting Future Mean Temperature

[79]:
```
# new_d1_group = new_d1_group[["Month","Year",'NDVI_mean', "Temperature_Mean"]]
X_clf_major = new_d1_group.iloc[:,:-1]
y_clf_major = new_d1_group["Temperature_Mean"].to_numpy()
scaler = StandardScaler()
X_clf_major = scaler.fit_transform(X_clf_major)
X_test = scaler.transform(future_predictions.iloc[:,:-1])
clf_major_0 = LinearRegression()
clf_major_0.fit(X_clf_major[::4],y_clf_major[::4])

clf_major_1 = SVR(kernel = "poly", degree = 6)
clf_major_1.fit(X_clf_major[1::4],y_clf_major[1::4])

clf_major_2 = LinearRegression()
clf_major_2.fit(X_clf_major[2::4],y_clf_major[2::4])

clf_major_3 = LinearRegression()
clf_major_3.fit(X_clf_major[3::4],y_clf_major[3::4])

temp_tp_0 = clf_major_0.predict(X_clf_major[::4])
temp_tp_1 = clf_major_1.predict(X_clf_major[1::4])
temp_tp_2 = clf_major_2.predict(X_clf_major[2::4])
temp_tp_3 = clf_major_3.predict(X_clf_major[3::4])

temp_test_0 = clf_major_0.predict(X_test[::4])
temp_test_1 = clf_major_1.predict(X_test[1::4])
temp_test_2 = clf_major_2.predict(X_test[2::4])
temp_test_3 = clf_major_3.predict(X_test[3::4])

temp_train_predictions = []
temp_test_predictions  = []

for i in range(19):
    temp_train_predictions.append(temp_tp_0[i])
    temp_train_predictions.append(temp_tp_1[i])
    temp_train_predictions.append(temp_tp_2[i])
    temp_train_predictions.append(temp_tp_3[i])
```

```
for i in range(4):
    temp_test_predictions.append(temp_test_0[i])
    temp_test_predictions.append(temp_test_1[i])
    temp_test_predictions.append(temp_test_2[i])
    temp_test_predictions.append(temp_test_3[i])

# clf_major.score(X_clf_major.to_numpy(),y_clf_major)
temp_r2_score = r2_score(new_d1_group["Temperature_Mean"],temp_train_predictions)
print("The R2 score for this Temperature model = ",temp_r2_score)

plt.figure(figsize = (18,8))
plt.plot(new_d1_group.index,temp_train_predictions, lw = 5, label = "Train Predictions")
plt.plot(new_d1_group.index,new_d1_group["Temperature_Mean"], label = "Original Values")
plt.plot(future_predictions.index, temp_test_predictions, label = "Future Predictions", lw = 3)
# plt.plot(new_d1_group.index, clf_major.predict(X_clf_major), lw = 5, label = "Train Predictions")
# plt.plot(new_d1_group.index, y_clf_major, label = "Original Values")
# plt.plot(future_predictions.index, clf_major.predict(scaler.transform(future_predictions.iloc[:,:-1])), label = "Future Predictions")
plt.title("Mean Temperature Regression Curve For Bangalore", fontsize = 25)
plt.ylabel("Mean Temperature", fontsize = 20)
plt.axhline(y= 300, color='r', linestyle='--', lw=2 , label = "300 K")
plt.xlabel("Years ---->", fontsize = 20)
plt.legend(fontsize = 15)
plt.setp(plt.gca().get_yticklabels(), fontsize=15)
plt.setp(plt.gca().get_xticklabels(), fontsize=15)
future_predictions["Temperature_Mean"] = temp_test_predictions
future_predictions
```

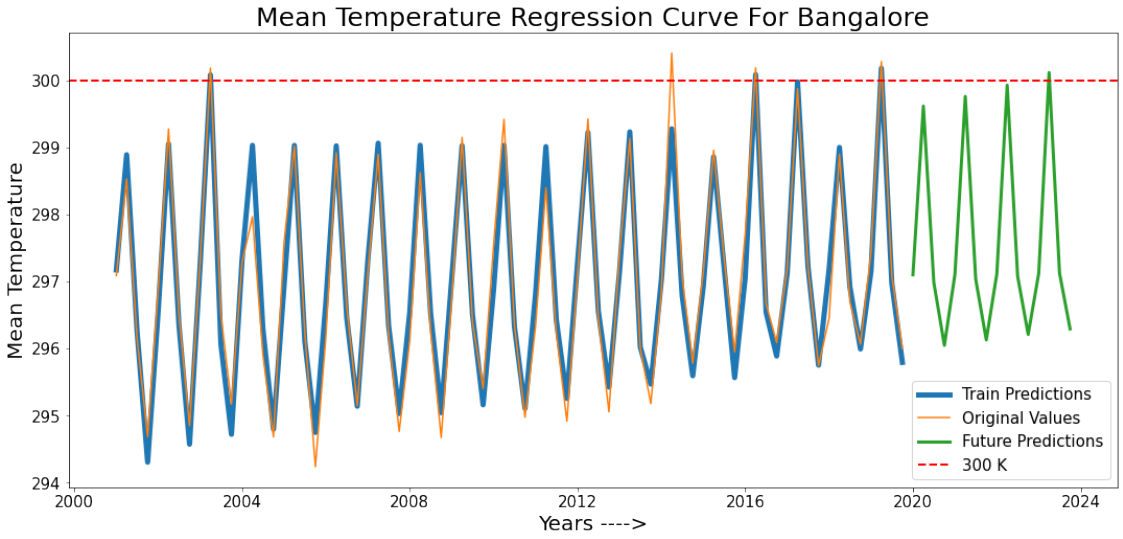
```
The R2 score for this Temperature model =  0.9511385775390162
```

[79]:

	Month	Year	AOD	EvapoTranspiration	NDVI_mean	\
2020-01-01	1	2020	522.355691	45.685520	2841.242598	
2020-04-01	4	2020	813.442515	48.346896	3226.741456	
2020-07-01	7	2020	504.522141	146.862655	2995.178019	
2020-10-01	10	2020	552.757363	119.759561	3566.398475	
2021-01-01	1	2021	528.501469	44.471675	2829.284085	
2021-04-01	4	2021	830.688162	45.574263	3225.278003	
2021-07-01	7	2021	505.883661	144.305758	2967.574291	
2021-10-01	10	2021	564.641124	117.097452	3529.355081	
2022-01-01	1	2022	534.647247	43.257831	2817.325571	
2022-04-01	4	2022	847.933808	42.801630	3223.814551	
2022-07-01	7	2022	507.245181	141.748861	2939.970563	

2022-10-01	10	2022	576.524885		114.435343	3492.311687
2023-01-01	1	2023	540.793025		42.043986	2805.367058
2023-04-01	4	2023	865.179455		40.028998	3222.351098
2023-07-01	7	2023	508.606701		139.191964	2912.366835
2023-10-01	10	2023	588.408646		111.773234	3455.268294

	Temperature_Mean
2020-01-01	297.102407
2020-04-01	299.617459
2020-07-01	296.987347
2020-10-01	296.049084
2021-01-01	297.110187
2021-04-01	299.762485
2021-07-01	297.033892
2021-10-01	296.130848
2022-01-01	297.117967
2022-04-01	299.929311
2022-07-01	297.080438
2022-10-01	296.212612
2023-01-01	297.125747
2023-04-01	300.117795
2023-07-01	297.126983
2023-10-01	296.294376



The Regression Curve fits reasonably well and we can see that Mean Temperature has been projected to increase in the upcoming years.

[80]:
```
future_predictions["Temperature_Mean in °C"] =␣
↪future_predictions["Temperature_Mean"] -273.15
```

```
future_predictions
```

[80]:

	Month	Year	AOD	EvapoTranspiration	NDVI_mean	\
2020-01-01	1	2020	522.355691	45.685520	2841.242598	
2020-04-01	4	2020	813.442515	48.346896	3226.741456	
2020-07-01	7	2020	504.522141	146.862655	2995.178019	
2020-10-01	10	2020	552.757363	119.759561	3566.398475	
2021-01-01	1	2021	528.501469	44.471675	2829.284085	
2021-04-01	4	2021	830.688162	45.574263	3225.278003	
2021-07-01	7	2021	505.883661	144.305758	2967.574291	
2021-10-01	10	2021	564.641124	117.097452	3529.355081	
2022-01-01	1	2022	534.647247	43.257831	2817.325571	
2022-04-01	4	2022	847.933808	42.801630	3223.814551	
2022-07-01	7	2022	507.245181	141.748861	2939.970563	
2022-10-01	10	2022	576.524885	114.435343	3492.311687	
2023-01-01	1	2023	540.793025	42.043986	2805.367058	
2023-04-01	4	2023	865.179455	40.028998	3222.351098	
2023-07-01	7	2023	508.606701	139.191964	2912.366835	
2023-10-01	10	2023	588.408646	111.773234	3455.268294	

	Temperature_Mean	Temperature_Mean in °C
2020-01-01	297.102407	23.952407
2020-04-01	299.617459	26.467459
2020-07-01	296.987347	23.837347
2020-10-01	296.049084	22.899084
2021-01-01	297.110187	23.960187
2021-04-01	299.762485	26.612485
2021-07-01	297.033892	23.883892
2021-10-01	296.130848	22.980848
2022-01-01	297.117967	23.967967
2022-04-01	299.929311	26.779311
2022-07-01	297.080438	23.930438
2022-10-01	296.212612	23.062612
2023-01-01	297.125747	23.975747
2023-04-01	300.117795	26.967795
2023-07-01	297.126983	23.976983
2023-10-01	296.294376	23.144376

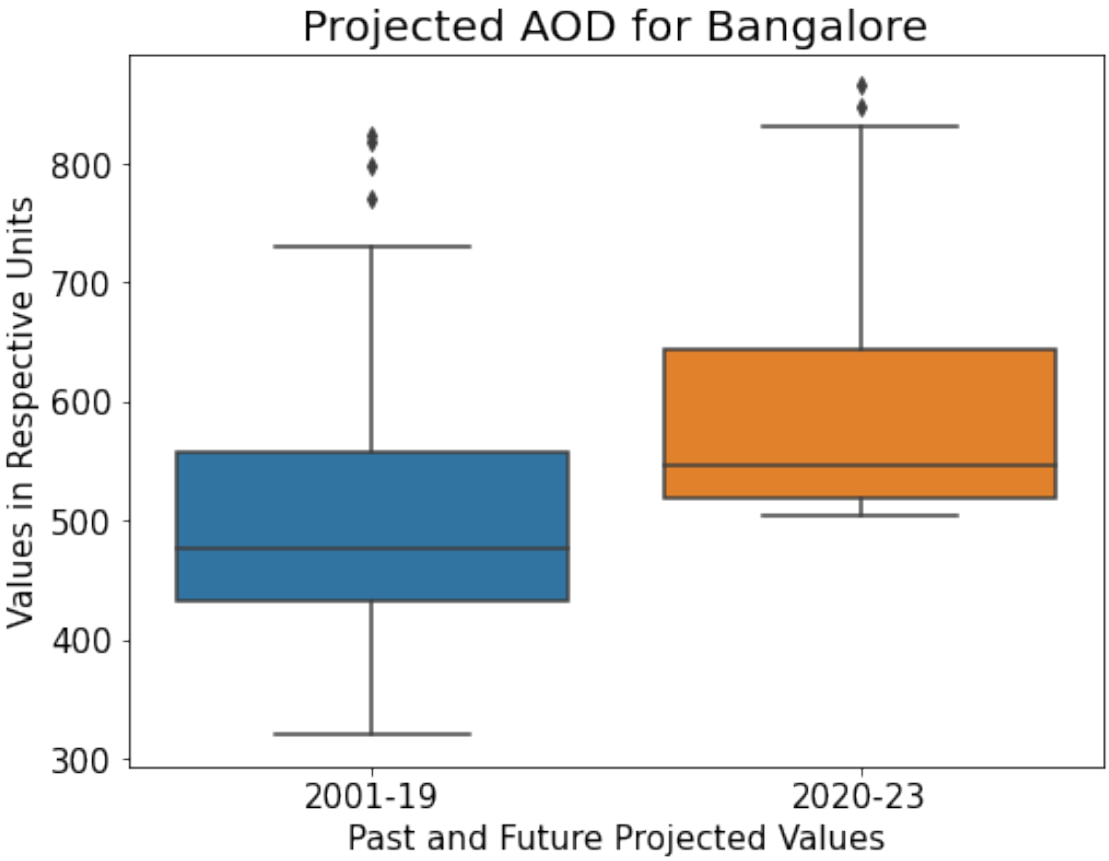
The Regression Curve fits relatively Well. We can observe that Mean Temperature has increased drastically over years by around 2° C from 2000. According to the current data available, the actual ground truth values of average Q1-2020, Q2-2020 and Q3-2020 temperature of Bangalore was 24.3° C, 26.6° C and 23.62° C Our model predicts the temperature to be 23.9° C, 26.4° C and 23.8° C. THese values are very close to the actual values and thus can be concluded that the model is able to predict the actual temperatures of Bangalore to a confidence of 0.005%.

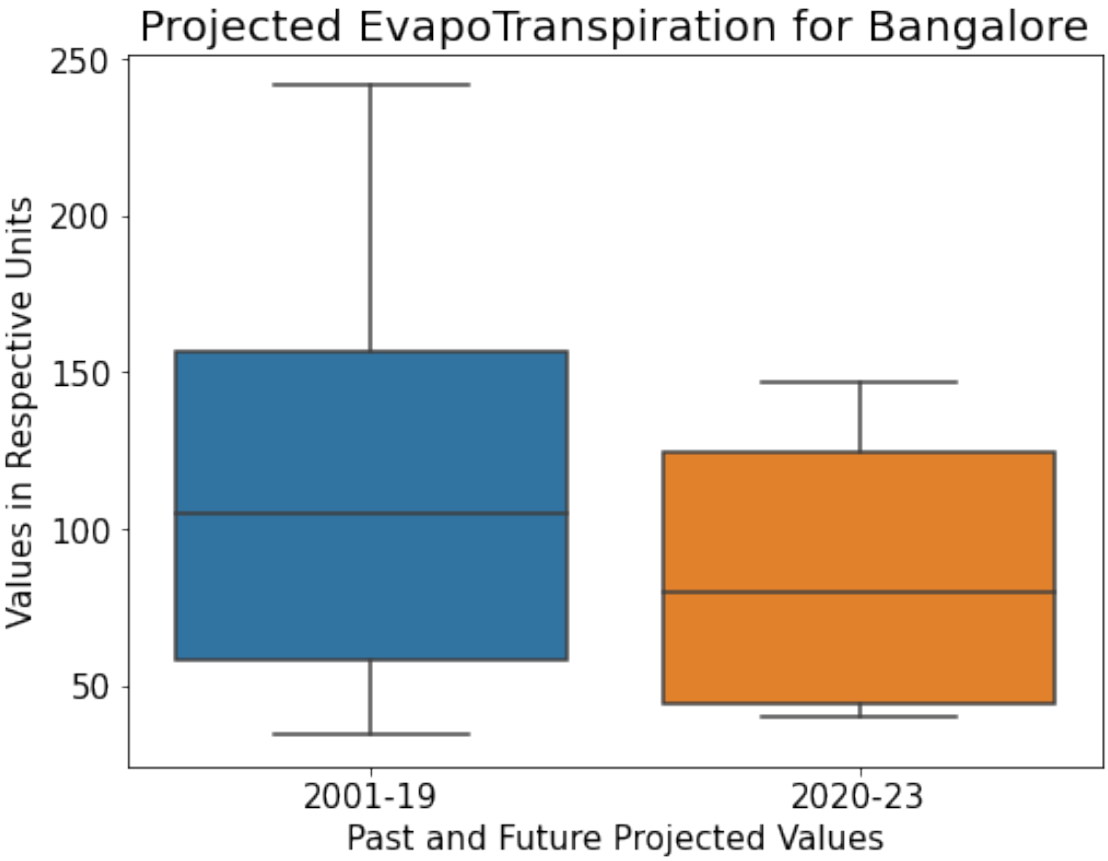
[81]:
```
future_predictions.describe().iloc[:,2:]
```

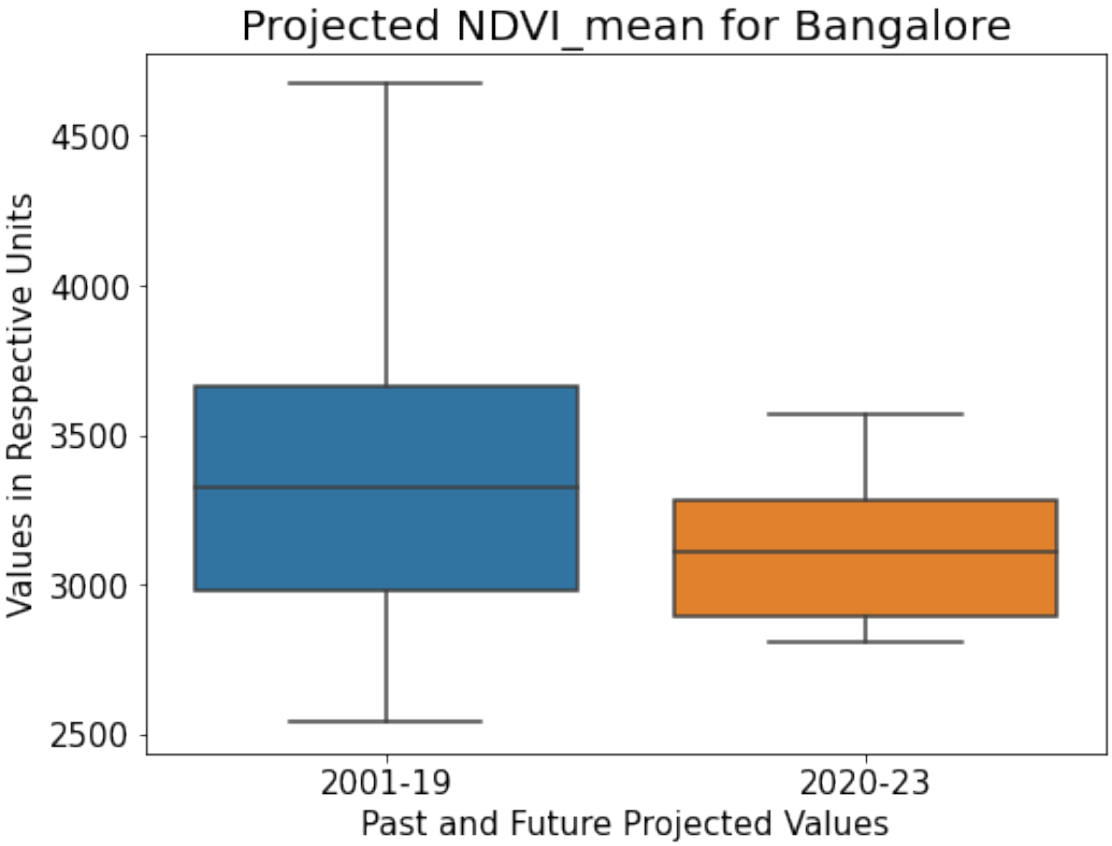
[81]:
```
                AOD  EvapoTranspiration    NDVI_mean  Temperature_Mean  \
count    16.000000           16.000000    16.000000         16.000000
mean    612.008192           86.711602  3128.114229        297.549934
std     138.148154           45.279092   274.190243          1.432920
min     504.522141           40.028998  2805.367058        296.049084
25%     518.918443           44.168214  2894.585776        296.814104
50%     546.775194           80.060065  3108.764559        297.106297
75%     644.667113          124.617662  3283.873166        297.749602
max     865.179455          146.862655  3566.398475        300.117795

       Temperature_Mean in °C
count               16.000000
mean                24.399934
std                  1.432920
min                 22.899084
25%                 23.664104
50%                 23.956297
75%                 24.599602
max                 26.967795
```

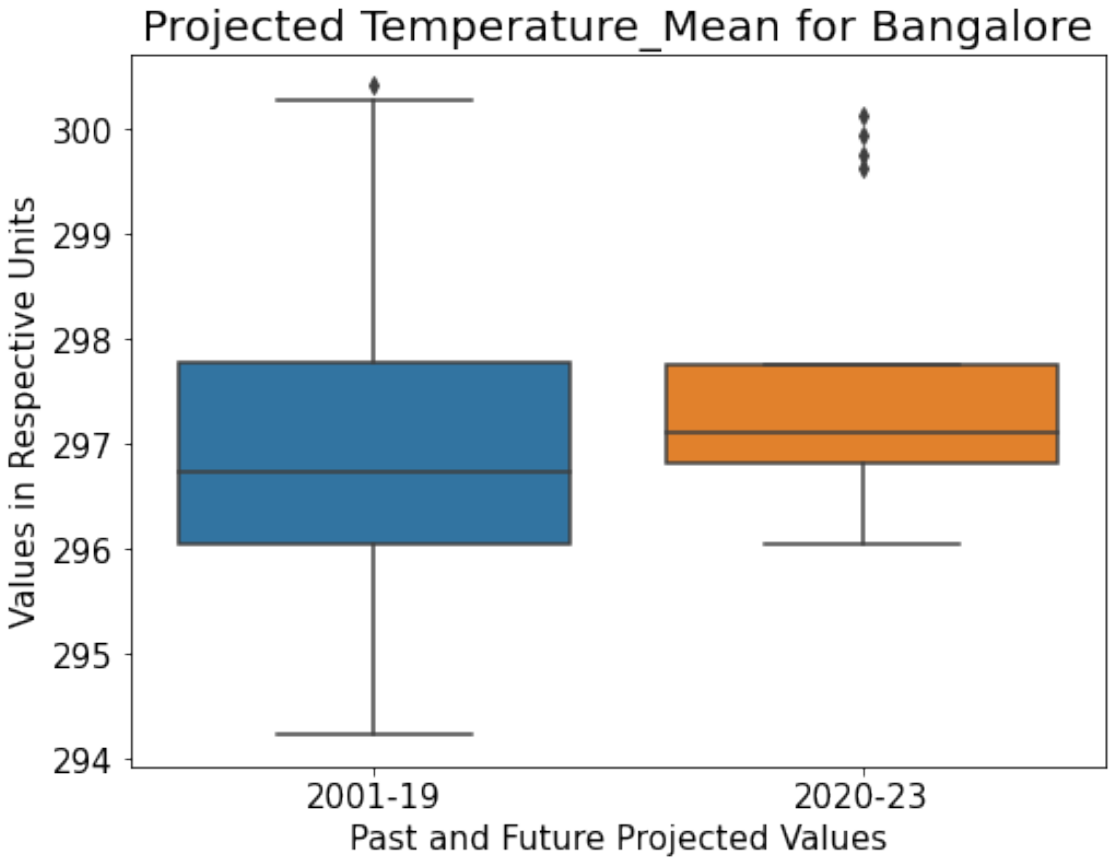
5.7 Comparison of Variables of Past and Future for Bangalore

[82]:
```
for i in future_predictions.columns[2:-1]:
    plt.figure(figsize = (8,6))
    sns.boxplot( data= [new_d1_group[i],future_predictions[i]])
    plt.title("Projected " +  i + " for Bangalore", fontsize = 20)
    xlabel = ["2001-19","2020-23"]
    plt.gca().set_xticklabels(xlabel)
    plt.setp(plt.gca().get_xticklabels(), fontsize=15)
    plt.gca().set_xlabel("Past and Future Projected Values", fontsize = 15)
    plt.setp(plt.gca().get_yticklabels(), fontsize=15)
    plt.gca().set_ylabel("Values in Respective Units", fontsize = 15)
    plt.show()
```


Projected AOD for Bangalore
Values in Respective Units
800
700
600
500
400
300
2001-19
2020-23
Past and Future Projected Values


Projected EvapoTranspiration for Bangalore
250
200
150
100
50
Values in Respective Units
2001-19
2020-23
Past and Future Projected Values


Projected NDVI_mean for Bangalore
Values in Respective Units
4500
4000
3500
3000
2500
2001-19
2020-23
Past and Future Projected Values



From the above plots, we can observe climate is kind of worsening in Bangalore. The AOD has spiked, EvapoTranspiration has decreased, The vegetation cover has also decreased, and this has led to increase in Temperature(albeit other unaccounted factors too must have influenced).

Conclusion

We can see that the Average Temperature has risen by around 1.25° C in the last 2 decades. This is a staunch indicator of the need to change certain policies and move on to renewable resources. Temperature rises are very evident these days in India. The evidences being very heavy rainfall in this year, raise of the temperature to abnormal levels in summers. One of the main reasons could be the pollution of air, deforestration and increased Green House effect due to the emmisions of the vehicals and industries. Thus poses a greater threat in future if neglected. Therefore necessary precautions and measures have to be taken to improve the condition of the environment. Necessary policies have to be implemented. To give our future generations a better place to live, we have to take the required steps.

The plight of Temperature of Bangalore has also been covered as a serious issue by several news articles.

https://www.thenewsminute.com/article/garden-city-warming-why-climate-change-must-be-election-issue-bengaluru-80176