A Project/Dissertation Report

on

Global warming Dataset analysis

Submitted in partial fulfillment of the
requirement for the award of the degree
of

B.tech-CSE

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Name of Supervisor:
Dr T. Ganesh sir

Submitted By
Yash Mishra 18SCSE1010525
Shantanu Chaurasia18SCSE1010677

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

OCTOBER,2021

Abstract

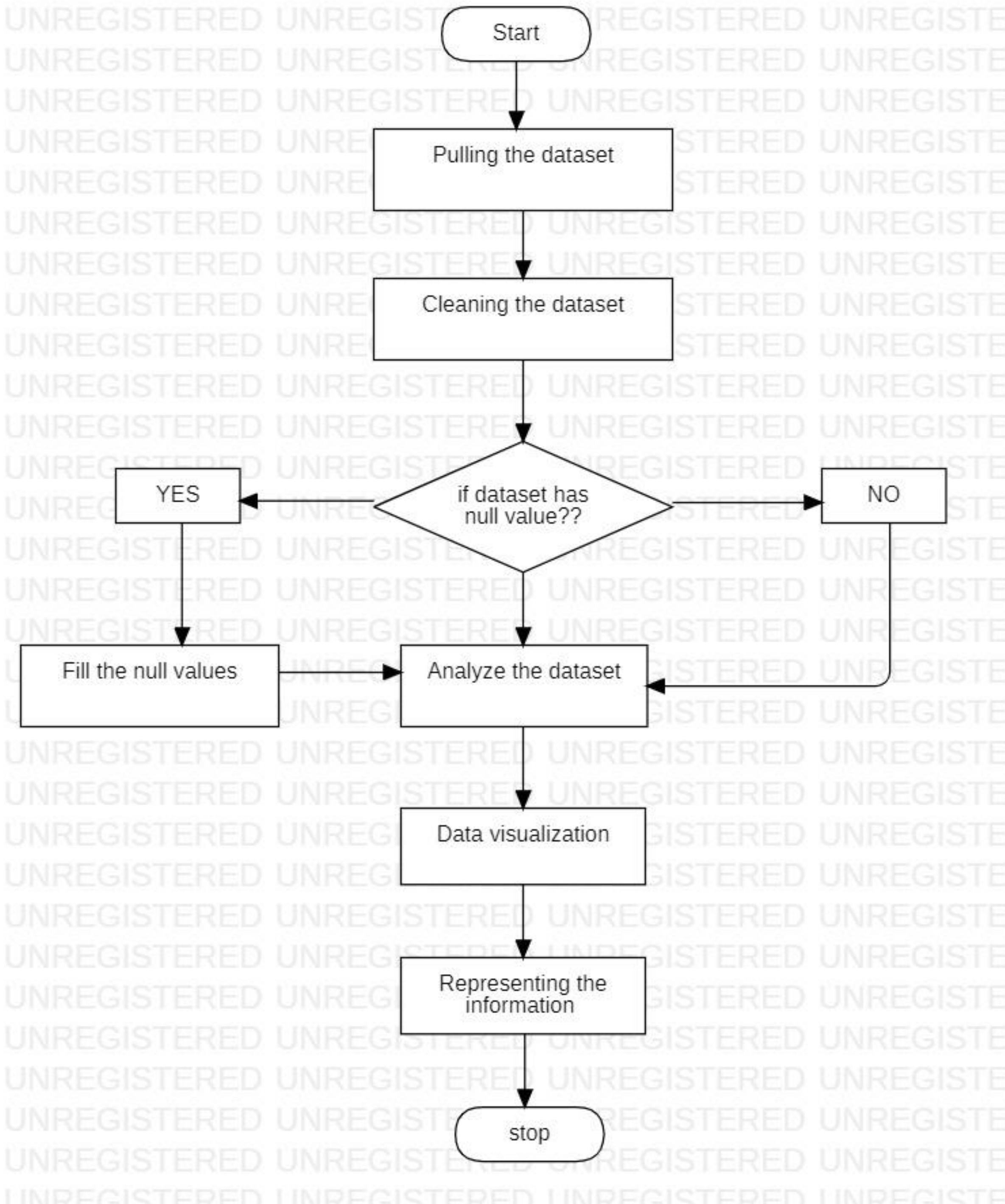
Global warming is a major concern nowadays. Weather conditions are changing, and it seems that human activity is one of the main causes. In fact, since the beginning of the industrial revolution, the burning of fossil fuels has increased the non-natural emissions of carbon dioxide to the atmosphere. Carbon dioxide is a greenhouse gas that absorbs the infrared radiation produced by the reflection of the sunlight on the Earth’s surface, trapping the heat in the atmosphere. Global warming and the associated climate changes are being the subject of intensive research due to their major impact on social, economic, and health aspects of human life. The combination of open scientific data, provided freely by reputable organizations such as NASA, and the numerous open-source tools make analyzing climate data accessible to everyone. Data scientists have the skills and expertise to transform raw data into knowledge and insights. In this project, we will be analyzing the global warming dataset which is time series data and we will extract insight from this data such as the temperature of different developed countries and different developing countries.

Student Data

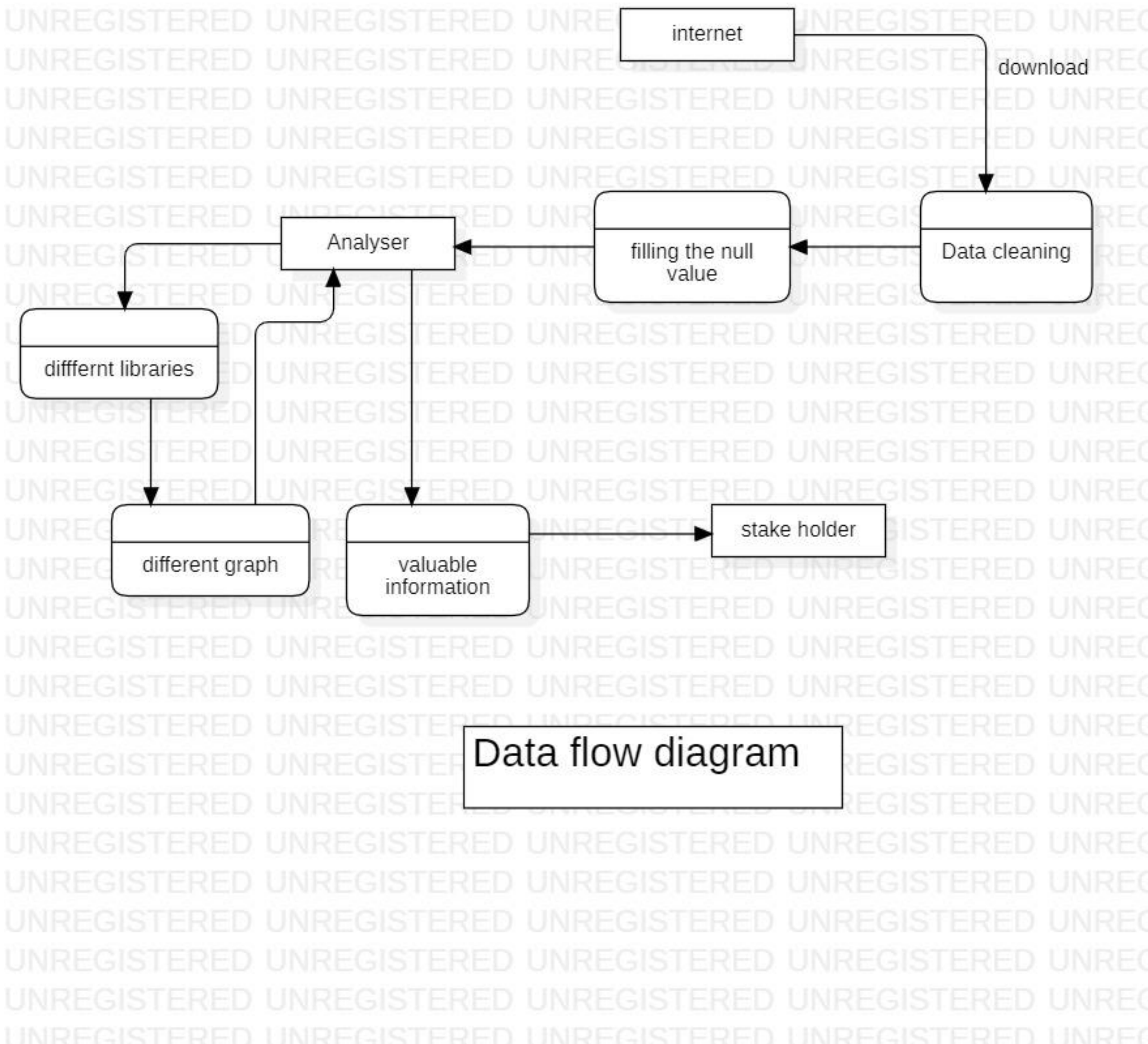
S.no	Name	Admission No	Enrollment No	Semester/year	department
1	Yash Mishra	18SCSE1010525	18021011753	7 / VI	B. TECH
2	Shantanu Chaurasia	18SCSE1010677	18021011900	7 / VI	B. TECH

List of Figures

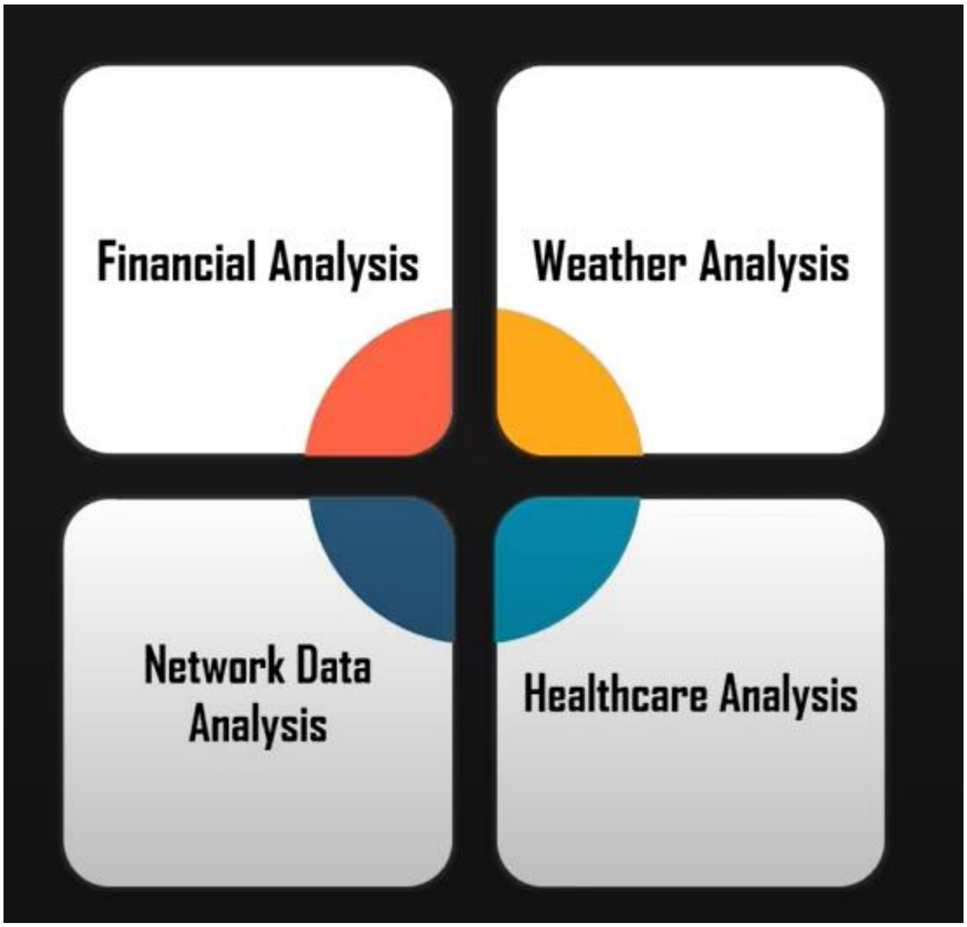
Flow Chart


Start
Pulling the dataset
Cleaning the dataset
if dataset has null value??
YES
NO
Fill the null values
Analyze the dataset
Data visualization
Representing the information
stop

Data Flow Diagram



Use Cases Time Series Analysis



Title

CHAPTER-1
Introduction

- Climate change is a change in the usual weather found in a place. This could be a change in how much rain a place usually gets in a year. Or it could be a change in a place's usual temperature for a month or season.

- People who study Earth see that Earth's climate is getting warmer. Earth's temperature has gone up about one degree Fahrenheit in the last 100 years. This may not seem like much. But small changes in Earth's temperature can have big effects.

- It is an undeniable fact that climate change possesses the biggest challenge for humanity in the current era. The global mean temp is constantly rising and is affecting the ocean, weather pattern, ice along with the planet, and animals. **Hence steps need to be taken to better understanding the climate changes and mitigate such harmful effects.**

- This can be done using effective climate data visualization and that’s exactly we are going to do in this project.

- We will do some simple climate modelling in a language called Python.

- We will be showing how to use both Python scripting and also spreadsheets to create simple models of things that go on in the earth system science.

Problem Statement

As we know climate is changing and due to that temperature is also increasing at an alarming rate. And because that, there are lots of social and economic impact, we see today Like sea level is rising due to the melting of ice because of that many of the people to leave their shelter. However, most of us did not aware how global warming have and will affect our life and at how the temperature increase in recent decades. So, we decide to analyze the dataset of global warming and represent in visualization because we human understand more clearly that way and it will help us understand very clearly how fast temperature is increasing

Tool and Technology

This is the Minimum requirement for this project:

- ❑ **HARDWARE REQUIREMENT:**

- The System needs minimum of 4 GB of Ram for smooth working of software.
- The system needs a minimum 1.3 GHz processor to run smoothly without any problem.

- ❑ **SOFTWARE REQUIREMENT:**

- Python IDLE
- Jupyter notebook
- Browser (Chrome, Edge)
- Google Colab

- ❑ **LANGUAGE:**

- Python

- ❑ LIBRARIES:

- NUMPY
- Pandas
- Matplotlib
- Seaborn

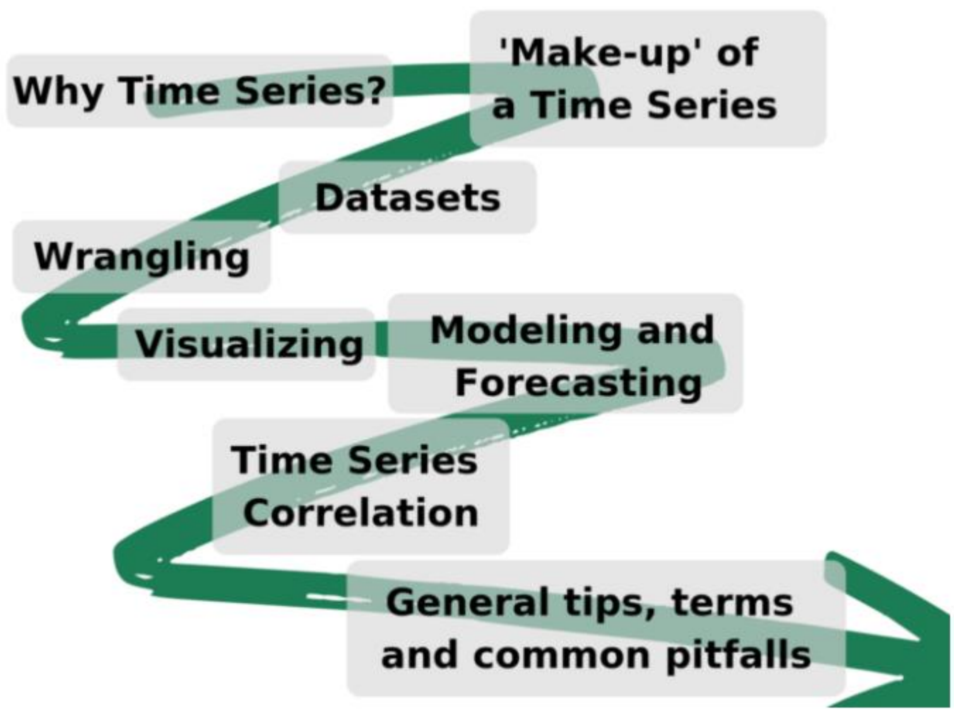
Why Time Series

All of life's scenes are placed in the foreground of time, take her away and there isn't a picture left that we can comprehend. Understanding time itself is not a pursuit for the faint-hearted, and we as humans are pretty much stuck comprehending time as a linear concept.

Time series analysis is useful for two major reasons:

- It allows us to understand and compare things without losing the important, shared background of 'time'
- It allows us to make forecasts

Workflow



Merits of the Project

- Understand what is Time Series Data.
- Helps in understanding Time Series Analysis.
- Helps in understanding Time Series Forcasting.
- Data Modeling.
- Analysize the overall average mean temperature of the globe and how it is changing with respect to time

Implementation

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import copy
%matplotlib inline

gt = pd.read_csv('C:/Users/my/Downloads/archive/GlobalTemperatures.csv', header=0,
index_col=0, parse_dates=True, squeeze=True)
gt.dropna(inplace = True)
gt.head()
```

dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature	LandMaxTemperatureUncertainty	LandMinTemperature	LandMinTemperatureUncertainty	LandAndOceanAverageTemperature	LandAndOcea
1850-01-01	0.749	1.105	8.242	1.738	-3.206	2.822	12.833	
1850-02-01	3.071	1.275	9.970	3.007	-2.291	1.623	13.588	
1850-03-01	4.954	0.955	10.347	2.401	-1.905	1.410	14.043	
1850-04-01	7.217	0.665	12.934	1.004	1.018	1.329	14.667	
1850-05-01	10.004	0.617	15.655	2.406	3.811	1.347	15.507	

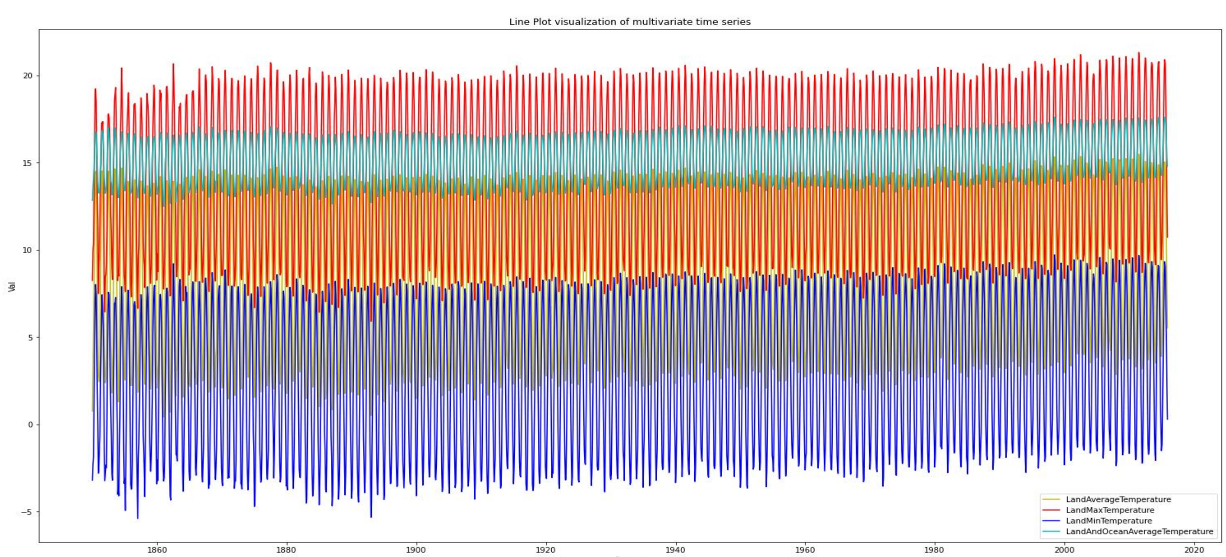
```
df = gt.reset_index(drop=True)
df
```

	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature	LandMaxTemperatureUncertainty	LandMinTemperature	LandMinTemperatureUncertainty	LandAndOceanAverageTemperature	LandAndOcean
0	0.749	1.105	8.242	1.738	-3.206	2.822	12.833	
1	3.071	1.275	9.970	3.007	-2.291	1.623	13.588	
2	4.954	0.955	10.347	2.401	-1.905	1.410	14.043	
3	7.217	0.665	12.934	1.004	1.018	1.329	14.667	
4	10.004	0.617	15.655	2.406	3.811	1.347	15.507	
...	...	...	...	...	...	...	...	
1987	14.755	0.072	20.699	0.110	9.005	0.170	17.589	
1988	12.999	0.079	18.845	0.088	7.199	0.229	17.049	
1989	10.801	0.102	16.450	0.059	5.232	0.115	16.290	
1990	7.433	0.119	12.892	0.093	2.157	0.106	15.252	
1991	5.518	0.100	10.725	0.154	0.287	0.099	14.774	

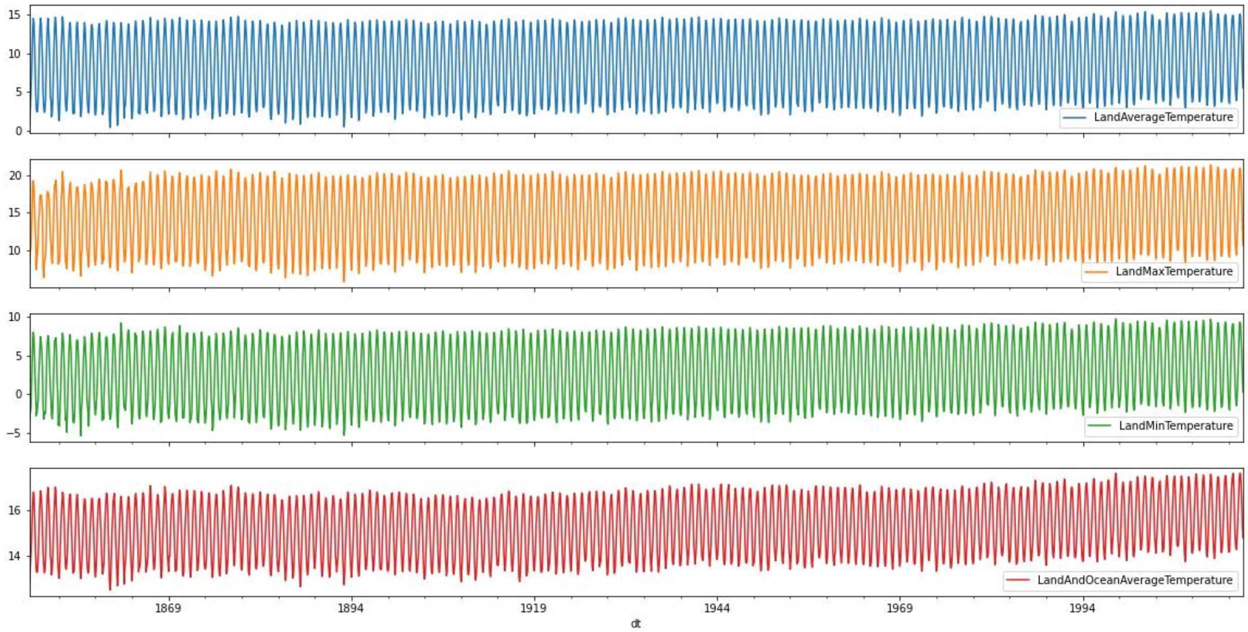
1992 rows × 8 columns

```
col = [gt.columns[0], gt.columns[2], gt.columns[4], gt.columns[6]]
col
Out[4]:
['LandAverageTemperature',
 'LandMaxTemperature',
 'LandMinTemperature',
 'LandAndOceanAverageTemperature']
```

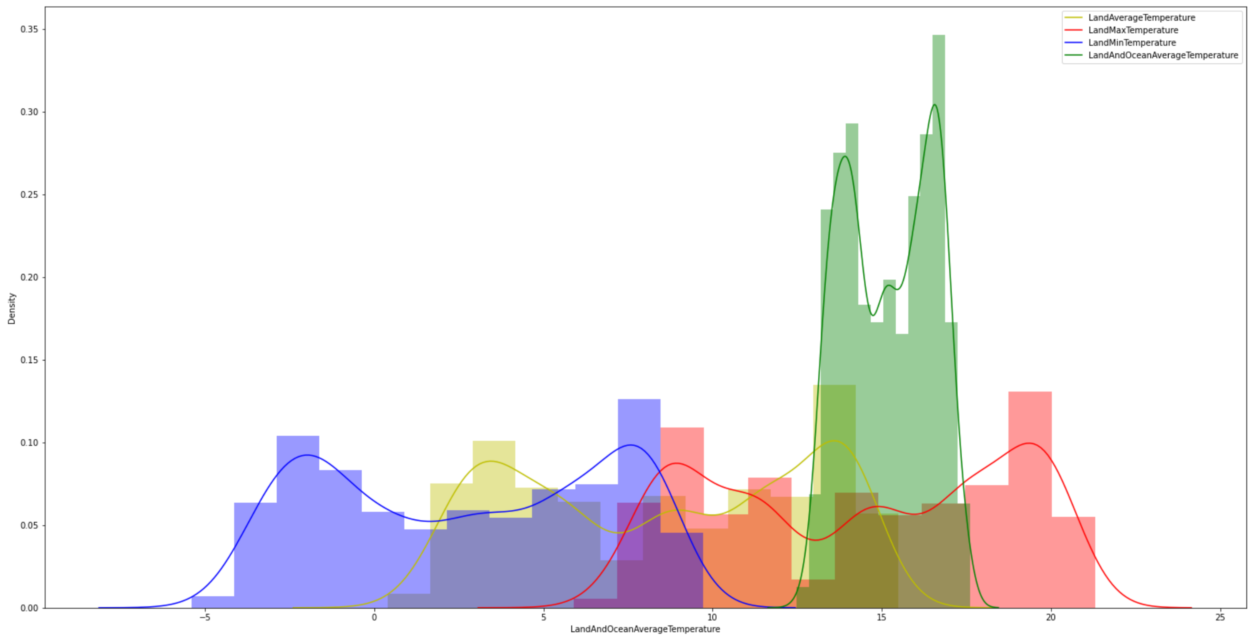
```
fig = plt.figure(figsize = (20, 10))
axes = fig.add_axes([0, 0, 1, 1])
axes.plot(col[0], data = gt, color = 'y')
axes.plot(col[1], data = gt, color = 'r')
axes.plot(col[2], data = gt, color = 'b')
axes.plot(col[3], data = gt, color = 'c')
axes.set_title('Line Plot visualization of multivariate time series')
axes.set_xlabel('Row no')
axes.set_ylabel('Val')
axes.legend()
fig.savefig('lineplot.png', bbox_inches = 'tight')
```



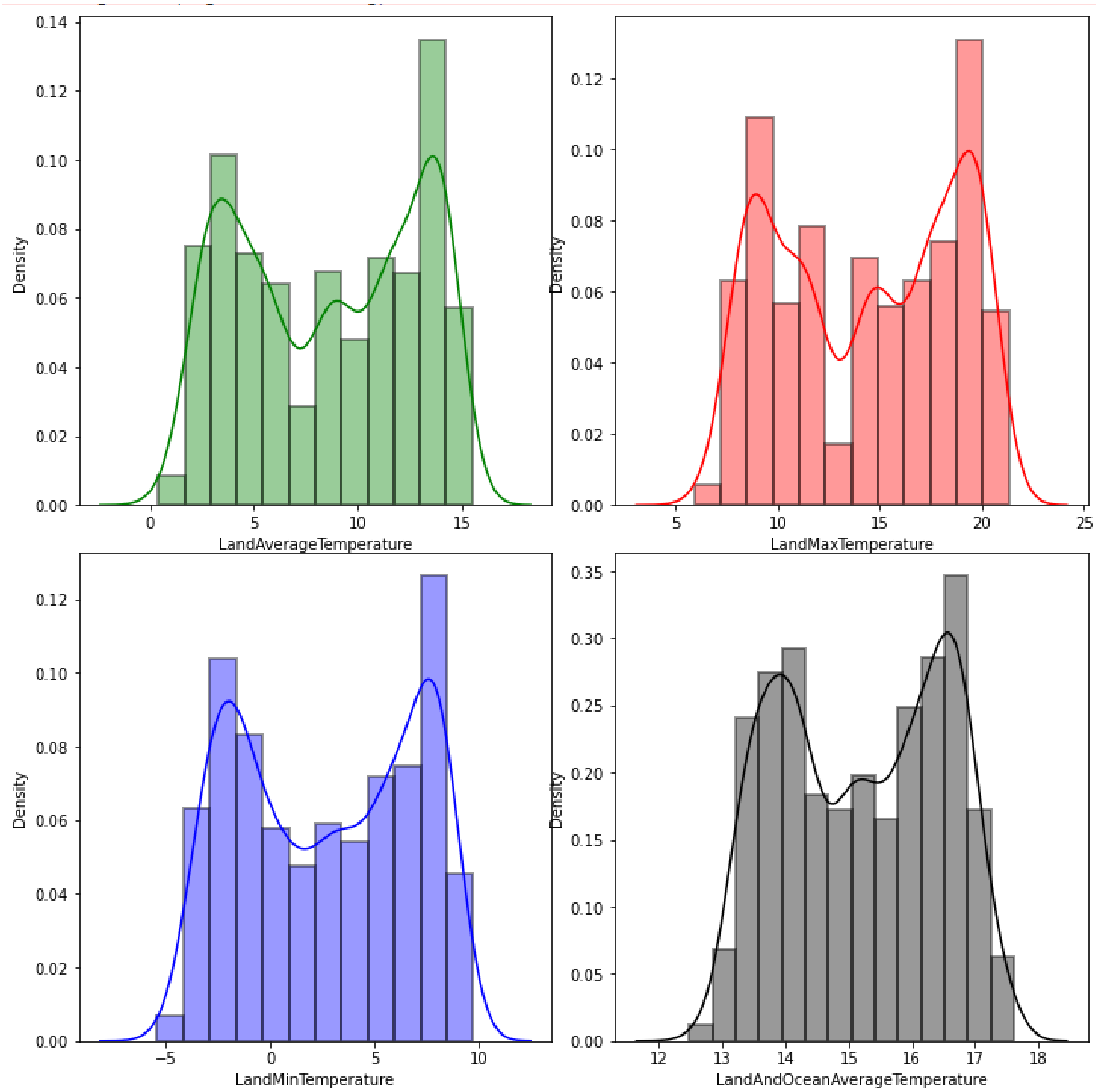
```
gt[col].plot(subplots=True, figsize=(20, 10))
plt.savefig('Linesubplots.png', bbox_inches = 'tight')
```



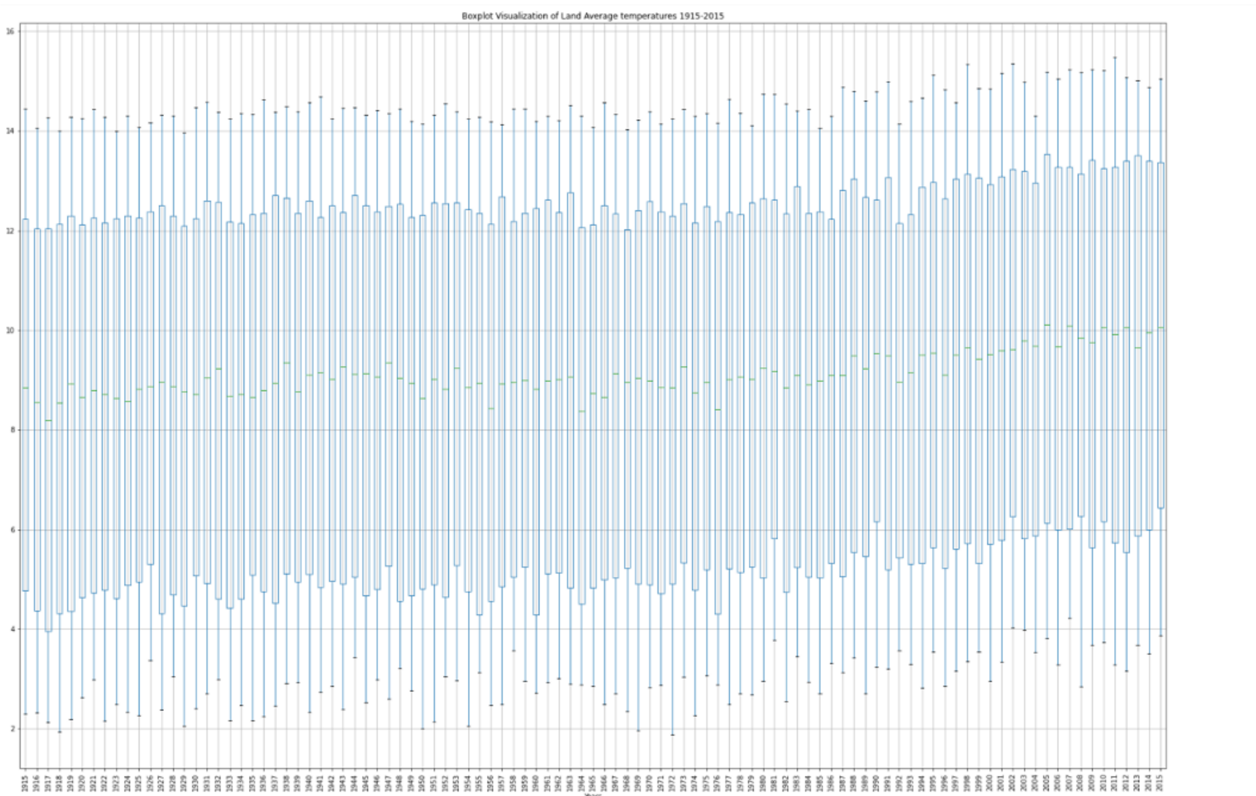

```
fig = plt.figure(figsize = (20, 10))
axes = fig.add_axes([0, 0, 1, 1])
sns.distplot(gt[col[0]], ax = axes, color = 'y')
sns.distplot(gt[col[1]], ax = axes, color = 'r')
sns.distplot(gt[col[2]], ax = axes, color = 'b')
sns.distplot(gt[col[3]], ax = axes, color = 'g')
axes.legend(col)
fig.savefig('distplot.png', bbox_inches = 'tight')
```




```
colors = [['g', 'r'], ['b', 'k']]
fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (10, 10))
plt.tight_layout()
data = np.reshape(col, (2, 2))

for i in range(2):
    for j in range(2):
        sns.distplot(gt[data[i][j]], ax = axes[i][j], hist_kws=dict(edgecolor= 'k', linewidth=2),
color = colors[i][j])

fig.savefig('distsubplot.png', bbox_inches = 'tight')
```



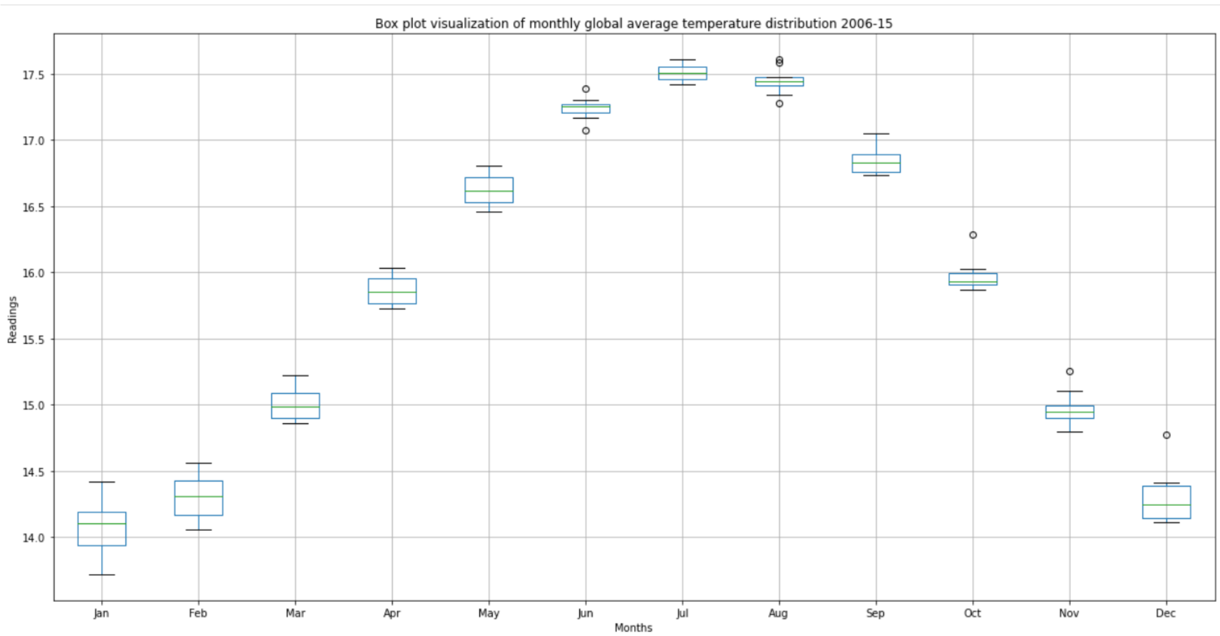

```
groups = gt[col[0]].groupby(pd.Grouper(freq='A'))
LandAverageTemperature = pd.DataFrame()
for name, group in groups:
   LandAverageTemperature[name.year] = group.values

LandAverageTemperature
LandAverageTemperature[LandAverageTemperature.columns[65:]].boxplot(figsize = (30, 20))
plt.xlabel('Years')
plt.title('Boxplot Visualization of Land Average temperatures 1915-2015')
plt.xticks(rotation = 90)
plt.savefig('boxplot.png', bbox_inches = 'tight')
```



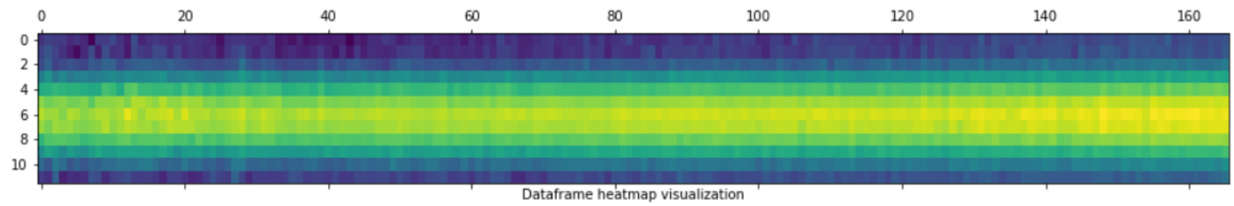

```
groups = gt[col[3]].groupby(pd.Grouper(freq='A'))
LandAndOceanAverageTemperature = pd.DataFrame()
for name, group in groups:
   if(name.year > 2005):
       LandAndOceanAverageTemperature[name.year] = group.values

# LandAndOceanAverageTemperature.columns
LandAndOceanAverageTemperature = LandAndOceanAverageTemperature.transpose()
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
LandAndOceanAverageTemperature.columns = months
LandAndOceanAverageTemperature
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2006	13.990	14.435	14.966	15.729	16.463	17.216	17.419	17.479	16.792	15.935	14.931	14.407
2007	14.417	14.408	15.017	15.930	16.629	17.168	17.485	17.339	16.737	15.867	14.821	14.110
2008	13.719	14.061	15.077	15.735	16.497	17.077	17.449	17.282	16.757	15.931	14.919	14.151
2009	14.091	14.267	14.873	15.819	16.571	17.260	17.578	17.427	16.864	15.910	14.968	14.298
2010	14.208	14.517	15.223	16.039	16.732	17.271	17.532	17.412	16.761	15.939	14.995	14.117
2011	13.928	14.193	14.880	15.832	16.523	17.203	17.568	17.475	16.762	15.873	14.799	14.198
2012	13.859	14.164	14.863	15.881	16.699	17.252	17.450	17.420	16.882	16.019	15.001	14.138
2013	14.117	14.359	14.952	15.749	16.609	17.257	17.503	17.462	16.894	15.905	15.107	14.339
2014	14.136	14.157	15.090	16.038	16.804	17.303	17.508	17.607	16.975	16.029	14.899	14.410
2015	14.255	14.564	15.193	15.962	16.774	17.390	17.611	17.589	17.049	16.290	15.252	14.774

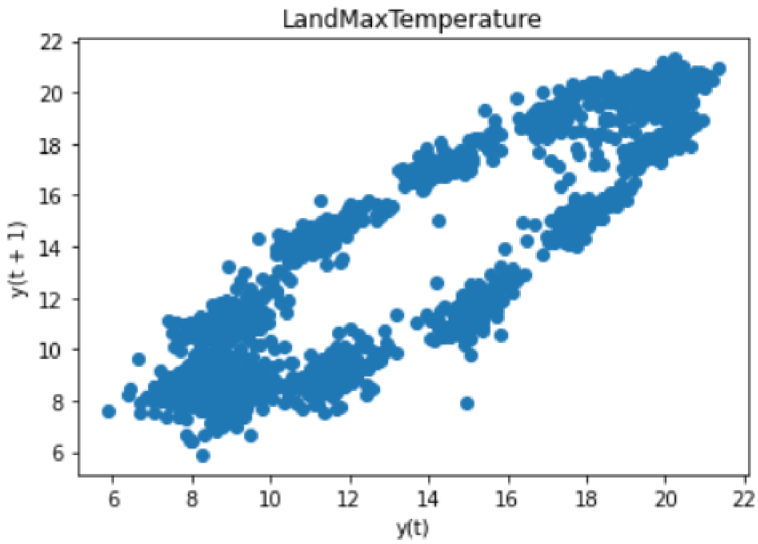
```
LandAndOceanAverageTemperature.boxplot(figsize = (20, 10))
plt.xlabel('Months')
plt.ylabel('Readings')
plt.title('Box plot visualization of monthly global average temperature distribution 2006-15')
plt.savefig('monthlyboxplot.png', bbox_inches = 'tight')
```



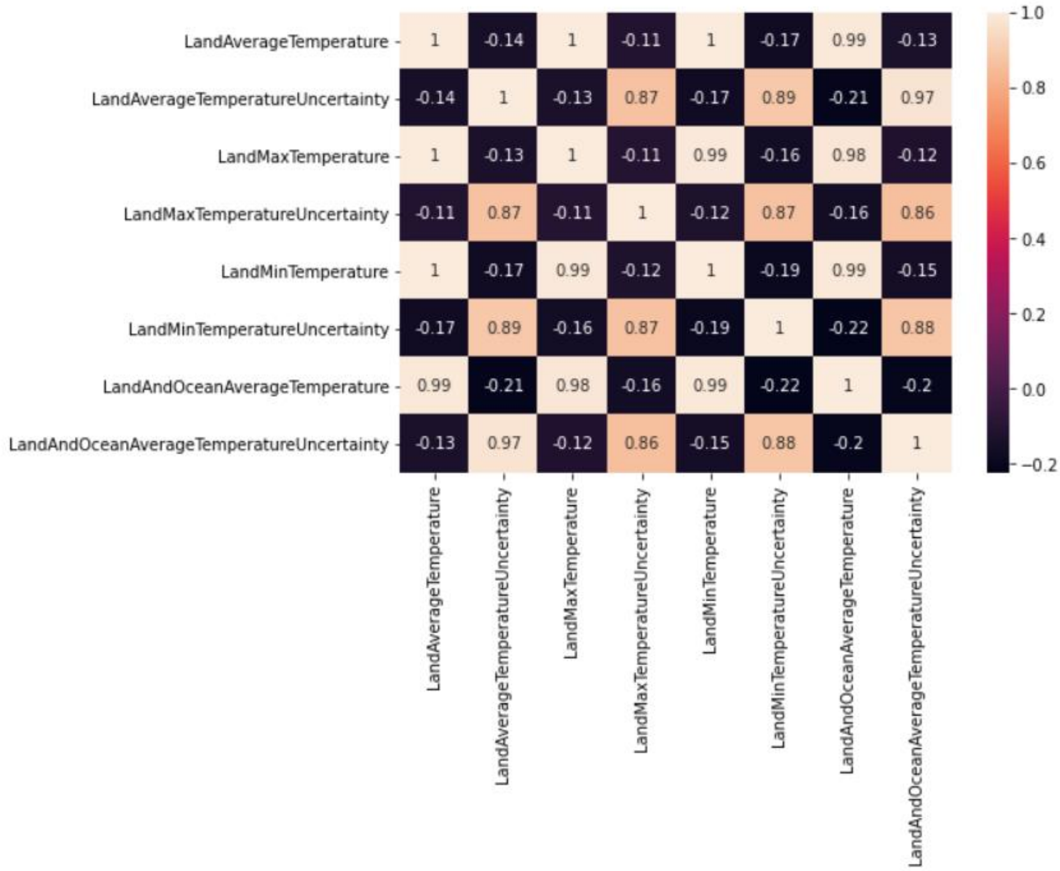
```
groups = gt[col[2]].groupby(pd.Grouper(freq='A'))
LandMinTemperature = pd.DataFrame()
for name, group in groups:
   LandMinTemperature[name.year] = group.values
# years = years.T
plt.matshow(LandMinTemperature, interpolation=None, aspect='auto')
plt.xlabel('Dataframe heatmap visualization')

plt.savefig('matviz.png', bbox_inches = 'tight')
```




```
pd.plotting.lag_plot(gt[col[1]])
plt.title('LandMaxTemperature')
plt.savefig('lagplot.png')
```




```
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
sns.heatmap(gt.corr(), annot=True)
fig.savefig('correlation_heatmap.png', bbox_inches = 'tight')
```




```
y = gt[col[0]]
y
```

```
dt
1850-01-01     0.749
1850-02-01     3.071
1850-03-01     4.954
1850-04-01     7.217
1850-05-01    10.004
               ...
2015-08-01    14.755
2015-09-01    12.999
2015-10-01    10.801
2015-11-01     7.433
2015-12-01     5.518
Name: LandAverageTemperature, Length: 1992, dtype: float64
```

In [17]:

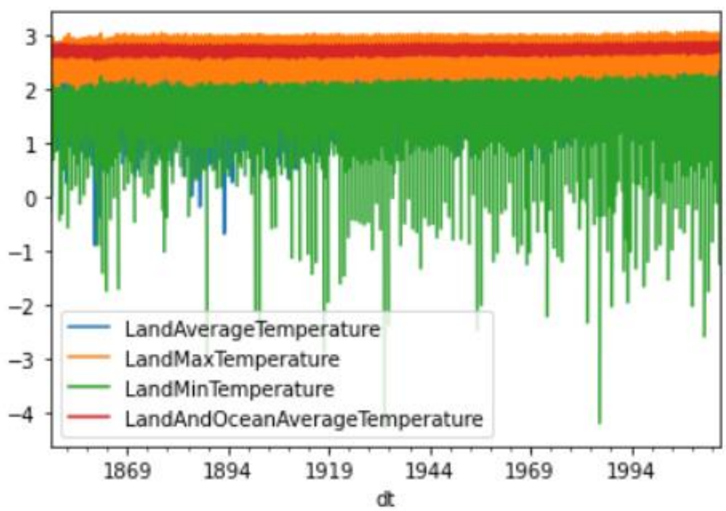
```
gt.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1992 entries, 1850-01-01 to 2015-12-01
Data columns (total 8 columns):
 #   Column                                     Non-Null Count  Dtype
---  ------                                     --------------  -----
 0   LandAverageTemperature                     1992 non-null   float64
 1   LandAverageTemperatureUncertainty          1992 non-null   float64
 2   LandMaxTemperature                         1992 non-null   float64
 3   LandMaxTemperatureUncertainty              1992 non-null   float64
 4   LandMinTemperature                         1992 non-null   float64
 5   LandMinTemperatureUncertainty              1992 non-null   float64
 6   LandAndOceanAverageTemperature             1992 non-null   float64
 7   LandAndOceanAverageTemperatureUncertainty  1992 non-null   float64
dtypes: float64(8)
memory usage: 140.1 KB
```

```
df1, df2 = gt[0:996], gt[996:]
m1, m2 = df1.mean(), df2.mean()
v1, v2 = df1.var(), df2.var()
mv = pd.DataFrame([m1, m2, v1, v2])
mv = mv.T
mv.columns = ['m1', 'm2', 'v1', 'v2']
mv
```

	m1	m2	v1	v2
LandAverageTemperature	8.226579	8.916586	18.727349	17.402247
LandAverageTemperatureUncertainty	0.426654	0.126672	0.050657	0.004732
LandMaxTemperature	14.058812	14.642390	18.736234	18.256921
LandMaxTemperatureUncertainty	0.804700	0.154863	0.465170	0.004067
LandMinTemperature	2.254853	3.232337	17.439677	16.641398
LandMinTemperatureUncertainty	0.693799	0.169899	0.255107	0.005262
LandAndOceanAverageTemperature	14.981343	15.443788	1.596759	1.544463
LandAndOceanAverageTemperatureUncertainty	0.180838	0.076226	0.004589	0.000769

```
gt1 = np.log(gt)
gt1[col].plot()
```

<AxesSubplot:xlabel='dt'>



city = pd.read_csv('C:/Users/my/Downloads/archive/GlobalLandTemperaturesByCity.csv')
country =
pd.read_csv('C:/Users/my/Downloads/archive/GlobalLandTemperaturesByCountry.csv')

city.head()

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
0	1743-11-01	6.068	1.737	Århus	Denmark	57.05N	10.33E
1	1743-12-01	NaN	NaN	Århus	Denmark	57.05N	10.33E
2	1744-01-01	NaN	NaN	Århus	Denmark	57.05N	10.33E
3	1744-02-01	NaN	NaN	Århus	Denmark	57.05N	10.33E
4	1744-03-01	NaN	NaN	Århus	Denmark	57.05N	10.33E

city.shape

Out[84]:

(8599212, 7)

In [85]:

```
country.head()
country.shape()
```

Out[85]:

```
(577462, 4)
```

In [86]:

```
city.Country.value_counts()
```

Out[86]:

```
India               1014906
China                827802
United States        687289
Brazil               475580
Russia               461234
                      ...
Namibia                1881
Djibouti               1797
Eritrea                1797
Oman                   1653
Papua New Guinea       1581
Name: Country, Length: 159, dtype: int64
```

In [87]:

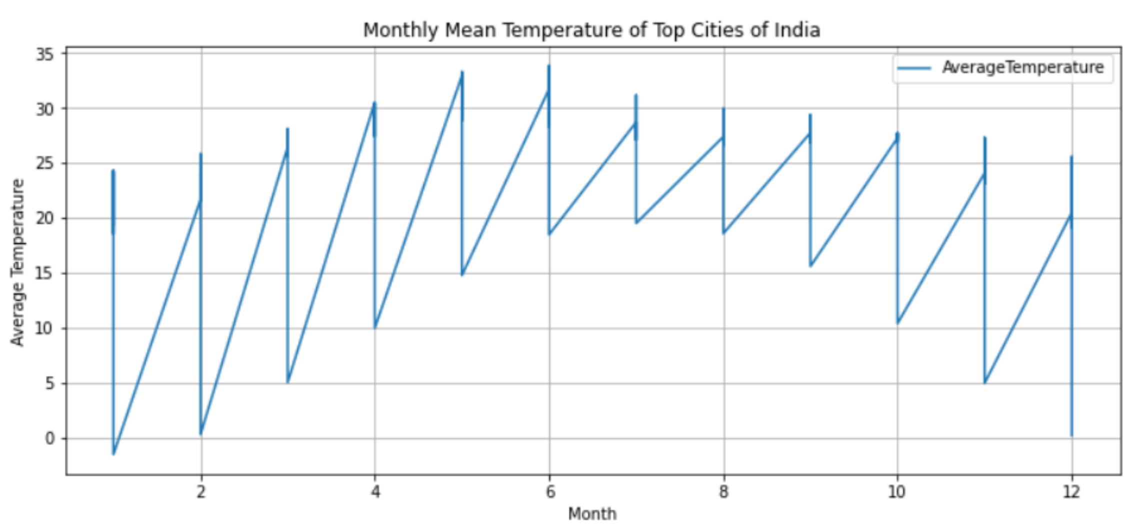
```
India = city[city.Country == 'India']
India.head()
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
49880	1816-03-01	19.934	2.258	Abohar	India	29.74N	73.85E
49881	1816-04-01	26.641	3.398	Abohar	India	29.74N	73.85E
49882	1816-05-01	32.535	2.408	Abohar	India	29.74N	73.85E
49883	1816-06-01	33.254	2.123	Abohar	India	29.74N	73.85E
49884	1816-07-01	31.105	1.848	Abohar	India	29.74N	73.85E

```
India = India.set_index('dt')
India.head()
```

dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
1816-03-01	19.934	2.258	Abohar	India	29.74N	73.85E
1816-04-01	26.641	3.398	Abohar	India	29.74N	73.85E
1816-05-01	32.535	2.408	Abohar	India	29.74N	73.85E
1816-06-01	33.254	2.123	Abohar	India	29.74N	73.85E
1816-07-01	31.105	1.848	Abohar	India	29.74N	73.85E

```
India_mean_temperature_monthly =
India.groupby([India.index.month.rename('Month'),India.City])['AverageTemperature'].me
an().reset_index()
India_mean_temperature_monthly.head()
top_cities =
India_mean_temperature_monthly[India_mean_temperature_monthly.City.isin(['Ahmadaba
d', 'Calcutta', 'Madras','New Delhi', 'Bombay', 'Srinagar'])]
top_cities = top_cities.set_index('Month')
top_cities.head()
top_cities.plot(figsize =(12,5))
plt.title('Monthly Mean Temperature of Top Cities of India')
plt.ylabel("Average Temperature")
plt.grid(True)
```



import random

import pandas **as** pd
import numpy **as** np

import matplotlib.pyplot **as** plt
import seaborn **as** sns

import datetime

import warnings

data =
pd.read_csv("C:/Users/my/Downloads/archive/GlobalLandTemperaturesByMajorCity.csv",
parse_dates=["dt"])
print(data.shape)
data.head()

```
(239177, 7)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
0	1849-01-01	26.704	1.435	Abidjan	Côte D'Ivoire	5.63N	3.23W
1	1849-02-01	27.434	1.362	Abidjan	Côte D'Ivoire	5.63N	3.23W
2	1849-03-01	28.101	1.612	Abidjan	Côte D'Ivoire	5.63N	3.23W
3	1849-04-01	26.140	1.387	Abidjan	Côte D'Ivoire	5.63N	3.23W
4	1849-05-01	25.427	1.200	Abidjan	Côte D'Ivoire	5.63N	3.23W

```
data.isna().sum()
```

```
dt                                   0
AverageTemperature               11002
AverageTemperatureUncertainty    11002
City                                 0
Country                              0
Latitude                             0
Longitude                            0
dtype: int64
```

checking the missing values

missing_values = data[data["AverageTemperature"].isna() == **True**]
print(missing_values.shape)
missing_values.head(5)

```
(11002, 7)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
36	1852-01-01	NaN	NaN	Abidjan	Côte D'Ivoire	5.63N	3.23W
37	1852-02-01	NaN	NaN	Abidjan	Côte D'Ivoire	5.63N	3.23W
38	1852-03-01	NaN	NaN	Abidjan	Côte D'Ivoire	5.63N	3.23W
39	1852-04-01	NaN	NaN	Abidjan	Côte D'Ivoire	5.63N	3.23W
40	1852-05-01	NaN	NaN	Abidjan	Côte D'Ivoire	5.63N	3.23W

```
missing_values["City"].value_counts()
```

```
Surabaya         386
Jakarta          386
Dar Es Salaam    379
Fortaleza        366
Karachi          289
                ...
Harbin             1
Izmir              1
Santiago           1
Nanjing            1
Wuhan              1
Name: City, Length: 98, dtype: int64
```

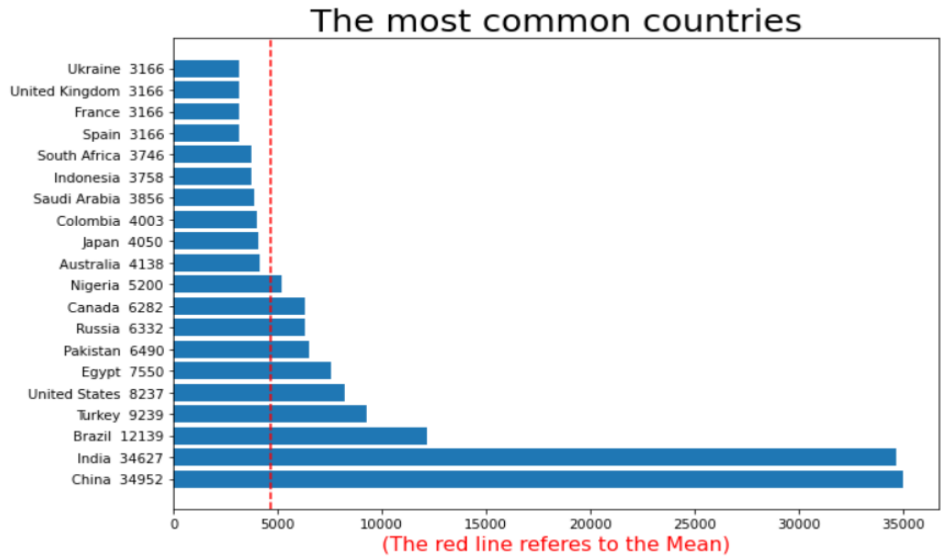
```
data.dropna(inplace=True)
data.shape
```

```
(228175, 7)
```

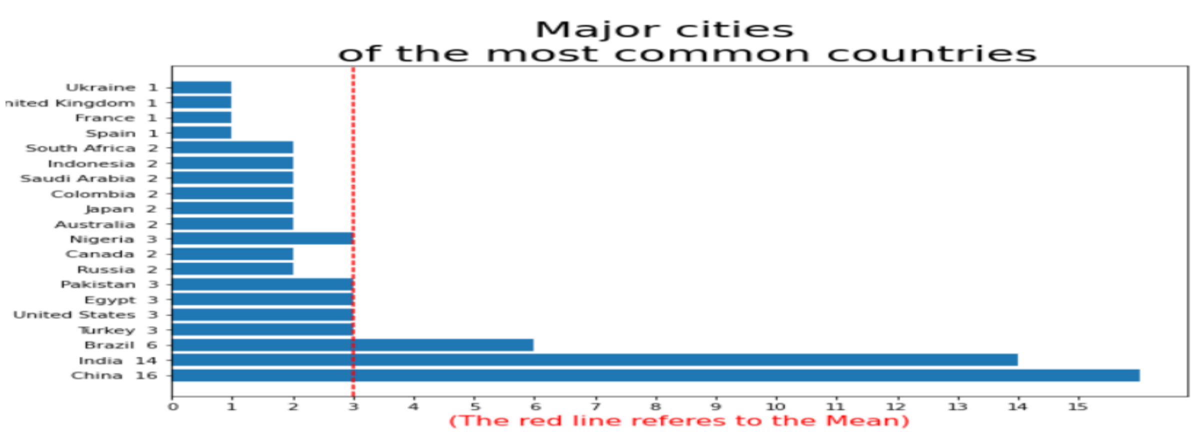
What are the most common Countries in our data set ?

```
most_countries = data["Country"].value_counts()[:20]
cwc = []
for i in zip(most_countries.index , most_countries.values):
    cit = i[0]+"  "+str(i[1])
    cwc.append(cit)
plt.figure(figsize=(10,7))
plt.barh(cwc , most_countries.values)
plt.axvline(x=data["Country"].value_counts().values.mean() , color="red" ,linestyle="--" )
plt.title("The most common countries" , fontsize=25)
plt.xlabel("(The red line referes to the Mean)" , c="red" , fontsize=15)
```

```
Text(0.5, 0, '(The red line referes to the Mean)')
```



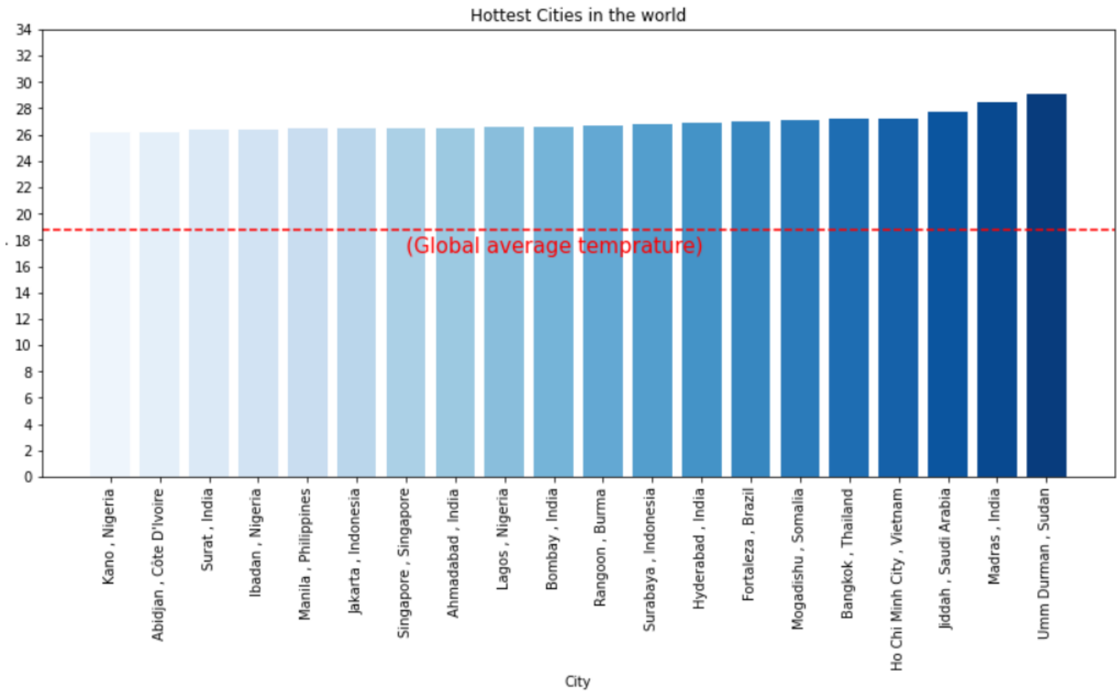

```
maj_count = []
cwc = []
for i in most_countries.index:
    temp = data[data["Country"] == i]["City"]
    maj_count.append(len(temp.unique()))
    cit = i+"  "+str(len(temp.unique()))
    cwc.append(cit)
plt.figure(figsize=(10,7))
plt.barh(cwc , maj_count)
plt.axvline(x=(int(sum(maj_count)/len(maj_count))) , color="red" ,linestyle="--" )
tex = "Major cities  \n of the most common countries"
plt.title(tex, fontsize=25 )
plt.xlabel("(The red line referes to the Mean)" , c="red" , fontsize=15)
xticks = plt.xticks(range(16))
```



```
temp = data.groupby(["Country" , "City"]).mean()
hottest = temp.sort_values(["AverageTemperature"] , ascending=False)[:20]
hottest = hottest.sort_values(["AverageTemperature"] , ascending=True)
coldest = temp.sort_values(["AverageTemperature"] , ascending=True)[:20]
coldest = coldest.sort_values(["AverageTemperature"] , ascending=False)

hottest_index = []
for i in hottest.index:
    cit = i[1] + " , " + i[0]
    hottest_index.append(cit)

coldest_index = []
for i in coldest.index:
    cit = i[1] + " , " + i[0]
    coldest_index.append(cit)
```

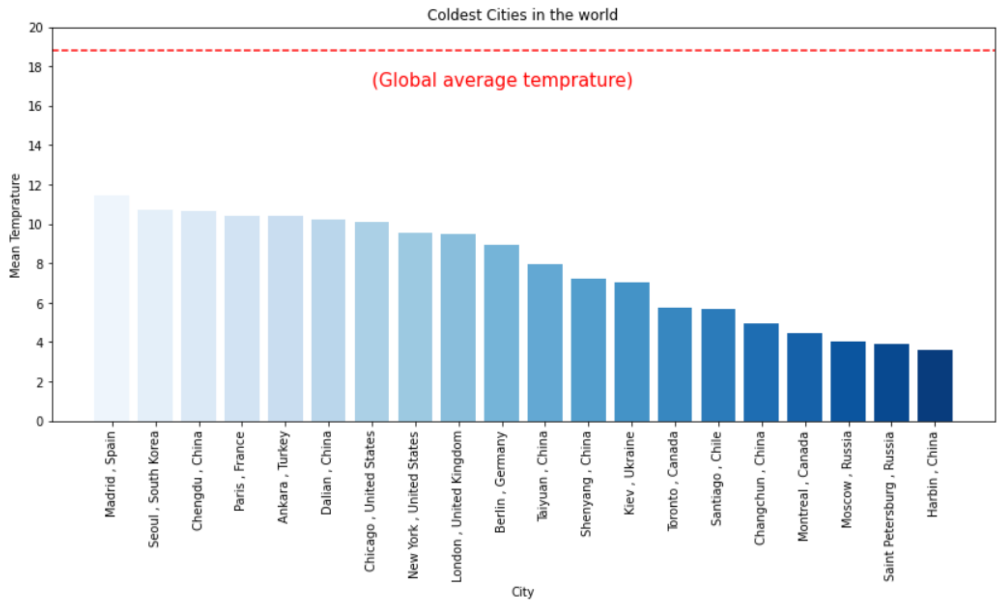
In [30]:

```
plt.figure(figsize=(14,6))

plt.bar(hottest_index , hottest.values[:,0]
        , color=sns.color_palette("Blues" , len(hottest) ))
plt.axhline(y=temp["AverageTemperature"].mean() , color="red" , linestyle="--")
plt.yticks(np.arange(0,35,2))
plt.xticks(rotation=90)
plt.xlabel("City")
plt.ylabel("Mean Temprature")
plt.title("Hottest Cities in the world")
plt.text(6,17,"(Global average temprature)" , color="red" , fontsize=15 )
```




```
plt.figure(figsize=(14,6))

plt.bar(coldest_index , coldest.values[:,0]
       , color=sns.color_palette("Blues" , len(coldest) ))
plt.axhline(y=temp["AverageTemperature"].mean() , color="red" , linestyle="--")
plt.yticks(np.arange(0,21,2))
plt.xticks(rotation=90)
plt.xlabel("City")
plt.ylabel("Mean Temprature")
plt.title("Coldest Cities in the world")
plt.text(6,17,"(Global average temprature)" , color="red" , fontsize=15 )
```

```
Text(6, 17, '(Global average temprature)')
```



Egypt

```
egypt_data = data[data["Country"] == "Egypt"]
egypt_data["Month"] = pd.DatetimeIndex(egypt_data["dt"]).month
egypt_data.drop(columns=["Country"] , axis=1 , inplace=True)
egypt_data.head(5)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Latitude	Longitude	Month
9224	1791-05-01	20.772	1.848	Alexandria	31.35N	30.16E	5
9225	1791-06-01	24.029	1.945	Alexandria	31.35N	30.16E	6
9226	1791-07-01	25.483	1.479	Alexandria	31.35N	30.16E	7
9227	1791-08-01	26.797	1.435	Alexandria	31.35N	30.16E	8
9228	1791-09-01	24.464	1.987	Alexandria	31.35N	30.16E	9

```
egypt_data["AverageTemperature"].describe()
```

```
count    7550.000000
mean       20.900406
std         5.265752
min         9.137000
25%        15.915250
50%        21.316500
75%        25.716250
max        30.767000
Name: AverageTemperature, dtype: float64
```

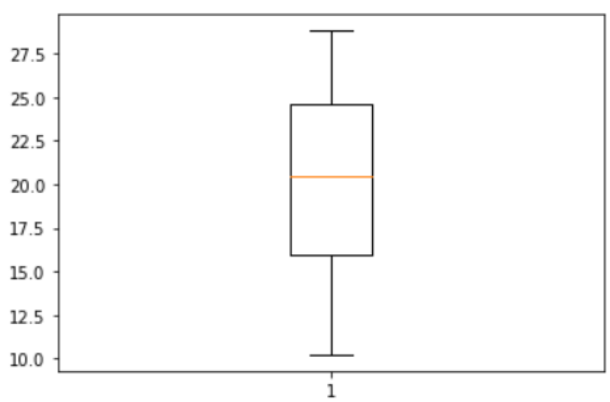
```
alex = egypt_data[egypt_data["City"] == "Alexandria"]
alex.drop(columns=["City" , "Latitude" , "Longitude"] , axis=1 , inplace=True)
alex.head(5)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Month
9224	1791-05-01	20.772	1.848	5
9225	1791-06-01	24.029	1.945	6
9226	1791-07-01	25.483	1.479	7
9227	1791-08-01	26.797	1.435	8
9228	1791-09-01	24.464	1.987	9

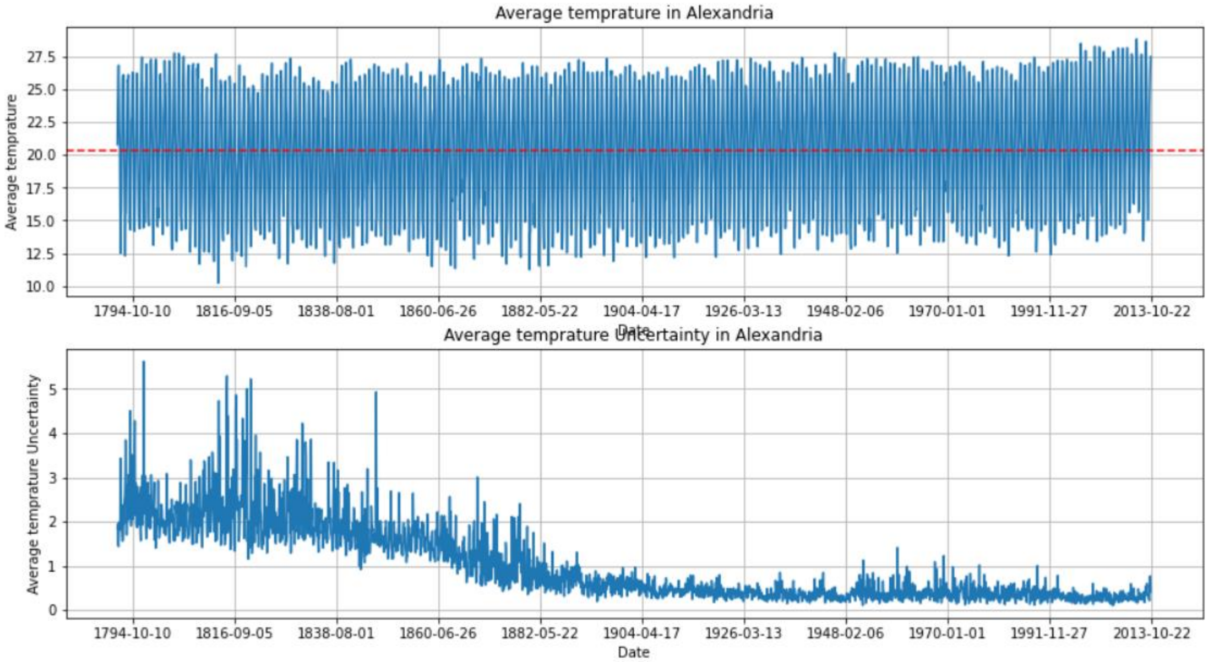
```
alex["AverageTemperature"].describe()
```

```
count    2666.000000
mean       20.312617
std         4.559545
min        10.227000
25%        15.987250
50%        20.463500
75%        24.612500
max        28.806000
Name: AverageTemperature, dtype: float64
```

```
fig = plt.boxplot(alex["AverageTemperature"])
```



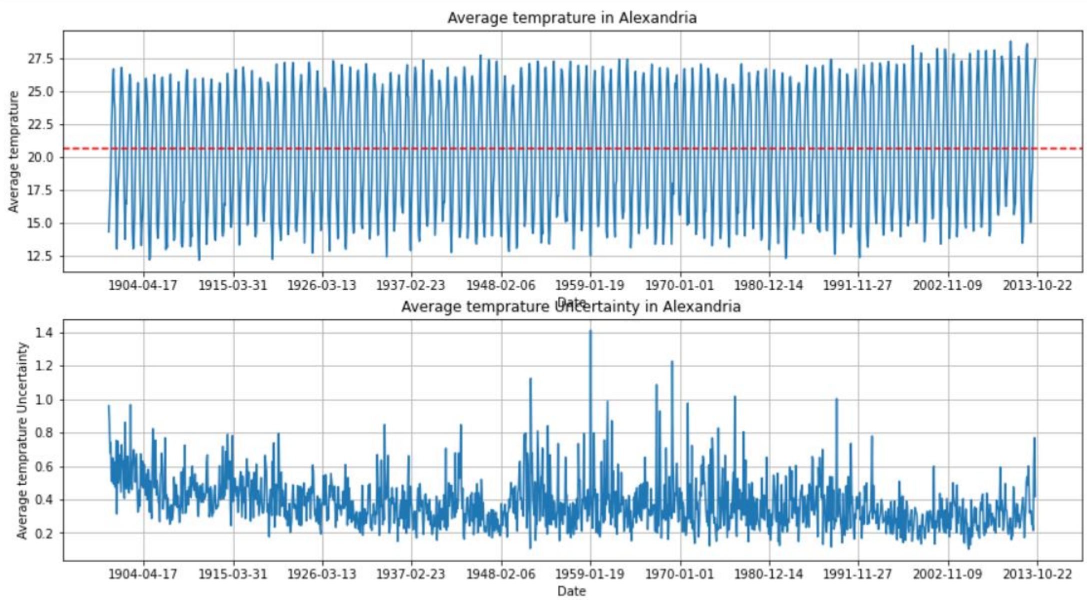

```
fig , ax = plt.subplots(2,figsize=(15,8))
ax[0].plot(alex["dt"] ,alex["AverageTemperature"])
ax[0].xaxis.set_major_locator(plt.MaxNLocator(12))
ax[0].axhline(y = alex["AverageTemperature"].mean() , color="red" , linestyle="--")
ax[0].set_title("Average temprature in Alexandria")
ax[0].set_xlabel("Date")
ax[0].set_ylabel("Average temprature")
ax[0].grid()

ax[1].plot(alex["dt"] ,alex["AverageTemperatureUncertainty"])
ax[1].xaxis.set_major_locator(plt.MaxNLocator(12))
#ax[0].axhline(y = alex["AverageTemperatureUncertainty"].mean() , color="red" , linestyle="--")
ax[1].set_title("Average temprature Uncertainty in Alexandria")
ax[1].set_xlabel("Date")
ax[1].set_ylabel("Average temprature Uncertainty")
ax[1].grid()
```




```
alex = alex[alex["dt"] >=  pd.Timestamp('1900-01-01 00:00:00')]
```

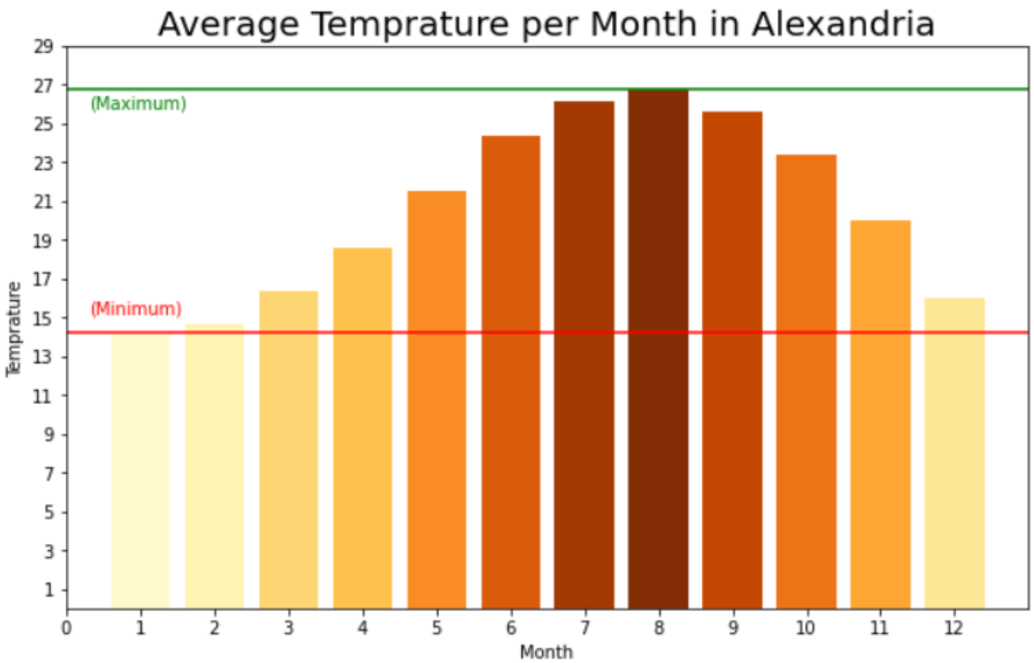
In [40]:

```
fig , ax = plt.subplots(2,figsize=(15,8))
ax[0].plot(alex["dt"] ,alex["AverageTemperature"])
ax[0].xaxis.set_major_locator(plt.MaxNLocator(12))
ax[0].axhline(y = alex["AverageTemperature"].mean() , color="red" , linestyle="--")
ax[0].set_title("Average temprature in Alexandria")
ax[0].set_xlabel("Date")
ax[0].set_ylabel("Average temprature")
ax[0].grid()

ax[1].plot(alex["dt"] ,alex["AverageTemperatureUncertainty"])
ax[1].xaxis.set_major_locator(plt.MaxNLocator(12))
#ax[0].axhline(y = alex["AverageTemperatureUncertainty"].mean() , color="red" ,
linestyle="--")
ax[1].set_title("Average temprature Uncertainty in Alexandria")
ax[1].set_xlabel("Date")
ax[1].set_ylabel("Average temprature Uncertainty")
ax[1].grid()
```




```
temp = alex.groupby(["Month"]).mean()
temp.drop(columns=["AverageTemperatureUncertainty"] , axis=1 , inplace=True)
temp = temp.sort_values(["AverageTemperature"])
plt.figure(figsize=(10,6))
plt.bar(temp.index , temp["AverageTemperature"].values ,
color=sns.color_palette("YlOrBr",len(temp.index) ))

plt.axhline(y=temp["AverageTemperature"].values.min() , color="red")
plt.text(.3,temp["AverageTemperature"].values.min()+1 , "(Minimum)" , color='red')
plt.axhline(y=temp["AverageTemperature"].values.max() , color="green")
plt.text(.3,temp["AverageTemperature"].values.max()-1 , "(Maximum)" , color='green')

xticks = plt.xticks(range(13))
yticks = plt.yticks(np.arange(1 , 30 ,2))
plt.xlabel("Month")
plt.ylabel("Temprature")
plt.title("Average Temprature per Month in Alexandria" , fontsize=20)
```

```
Text(0.5, 1.0, 'Average Temprature per Month in Alexandria')
```




```
cairo = egypt_data[egypt_data["City"] == "Cairo"]
cairo.drop(columns=["City" , "Latitude" , "Longitude"] , axis=1 , inplace=True)

#chossing only data after 1900
cairo = cairo[cairo["dt"] >=  pd.Timestamp('1900-01-01 00:00:00')]
cairo.head(5)
```

```
india_data = data[data["Country"] == "India"]
india_data["Month"] = pd.DatetimeIndex(india_data["dt"]).month
india_data.drop(columns=["Country"] , axis=1 , inplace=True)
india_data.head(5)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Latitude	Longitude	Month
3942	1796-01-01	19.649	2.286	Ahmadabad	23.31N	72.52E	1
3943	1796-02-01	21.632	1.770	Ahmadabad	23.31N	72.52E	2
3944	1796-03-01	24.953	2.427	Ahmadabad	23.31N	72.52E	3
3945	1796-04-01	30.297	1.827	Ahmadabad	23.31N	72.52E	4
3946	1796-05-01	33.223	1.496	Ahmadabad	23.31N	72.52E	5

```
india_data.tail()
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Latitude	Longitude	Month
216559	2013-04-01	30.546	0.279	Surat	21.70N	73.56E	4
216560	2013-05-01	32.980	1.097	Surat	21.70N	73.56E	5
216561	2013-06-01	29.418	0.527	Surat	21.70N	73.56E	6
216562	2013-07-01	27.306	0.257	Surat	21.70N	73.56E	7
216563	2013-08-01	27.187	0.129	Surat	21.70N	73.56E	8

```
india_data["AverageTemperature"].describe()
```

```
count    34627.000000
mean        25.809309
std          4.851196
min         11.378000
25%         22.935000
50%         26.518000
75%         29.254000
max         36.477000
Name: AverageTemperature, dtype: float64
```

```
delhi = india_data[india_data["City"] == "Delhi"]
delhi.drop(columns=["City" , "Latitude" , "Longitude"] , axis=1 , inplace=True)
delhi.head(5)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Month
63153	1796-01-01	14.590	2.374	1
63154	1796-02-01	17.109	1.940	2
63155	1796-03-01	21.454	2.608	3
63156	1796-04-01	28.715	2.122	4
63157	1796-05-01	33.726	1.997	5

```
delhi["AverageTemperature"].describe()
```

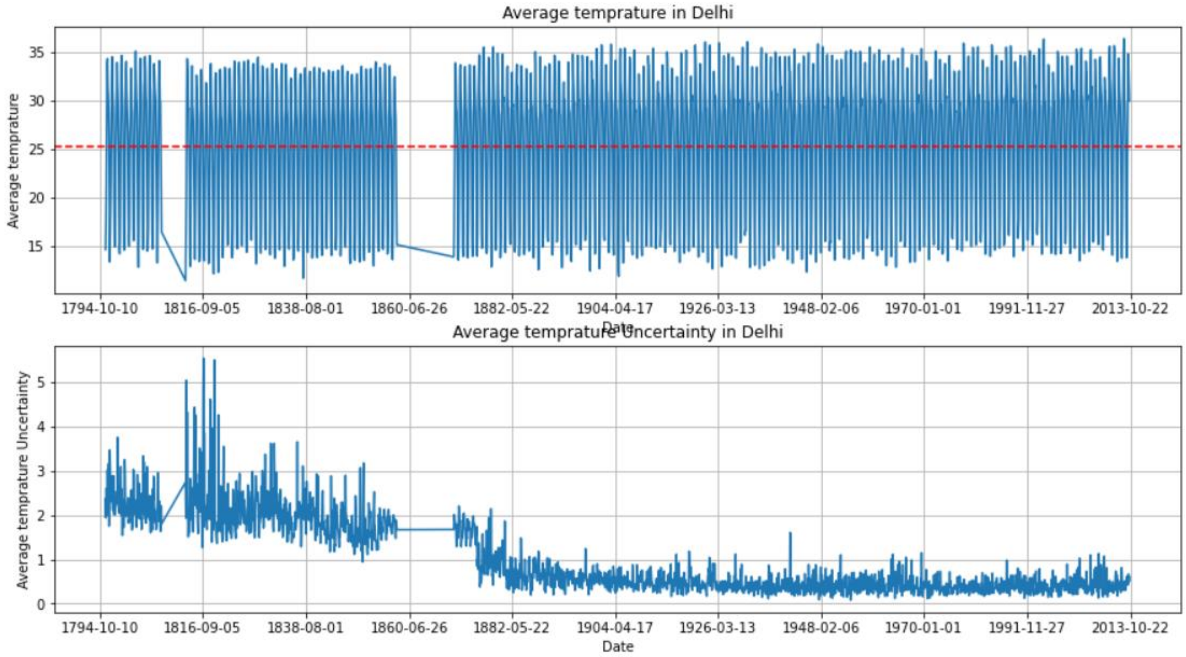
```
count    2394.000000
mean       25.165861
std         6.764657
min        11.378000
25%        19.058000
50%        27.151000
75%        30.592250
max        36.339000
Name: AverageTemperature, dtype: float64
```

```
fig = plt.boxplot(delhi["AverageTemperature"])
```

fig , dl = plt.subplots(2,figsize=(15,8))
dl[0].plot(delhi["dt"] ,delhi["AverageTemperature"])
dl[0].xaxis.set_major_locator(plt.MaxNLocator(12))
dl[0].axhline(y = delhi["AverageTemperature"].mean() , color="red" , linestyle="--")
dl[0].set_title("Average temprature in Delhi")
dl[0].set_xlabel("Date")

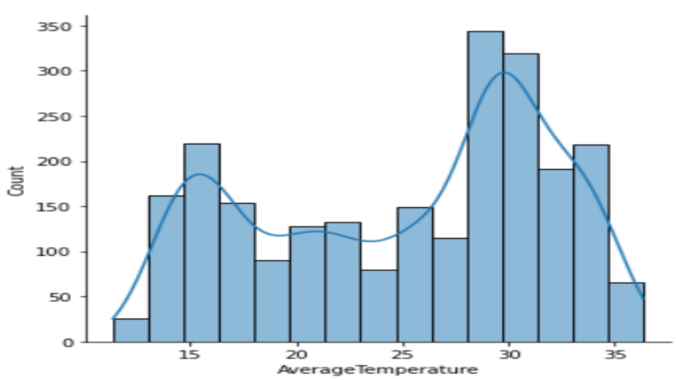
dl[0].set_ylabel("Average temprature")
dl[0].grid()

dl[1].plot(delhi["dt"] ,delhi["AverageTemperatureUncertainty"])
dl[1].xaxis.set_major_locator(plt.MaxNLocator(12))
#ax[0].axhline(y = delhi["AverageTemperatureUncertainty"].mean() , color="red" , linestyle="--")
dl[1].set_title("Average temprature Uncertainty in Delhi")
dl[1].set_xlabel("Date")
dl[1].set_ylabel("Average temprature Uncertainty")
dl[1].grid()



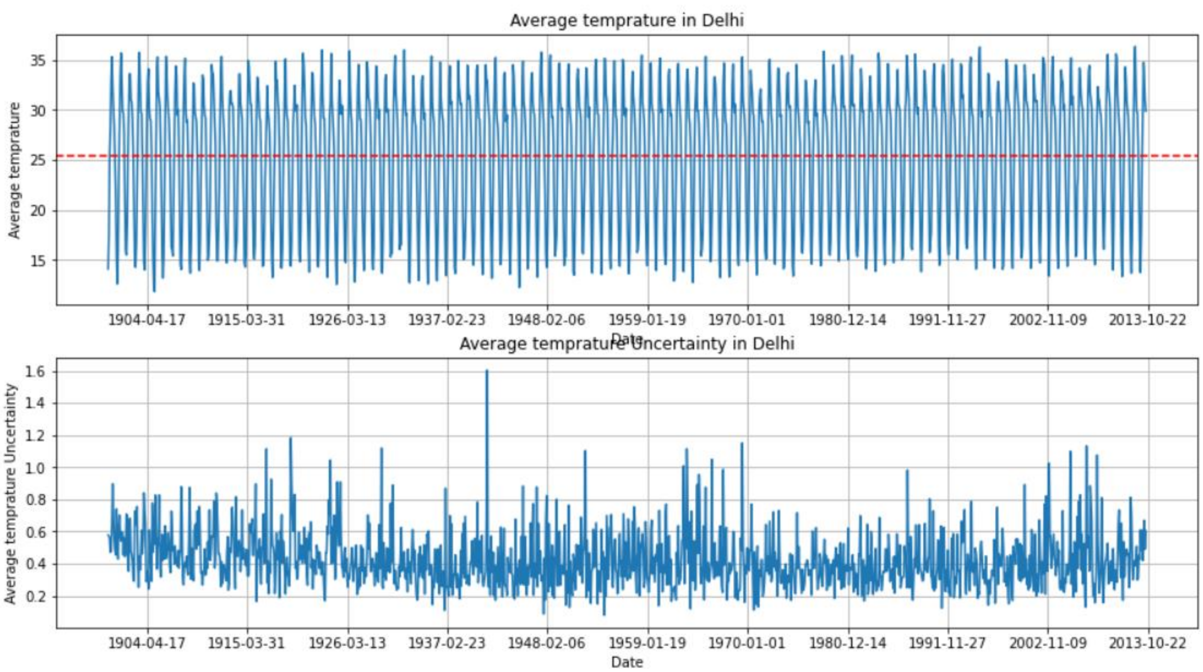

```
sns.displot(delhi["AverageTemperature"] , kde=True)
```

<seaborn.axisgrid.FacetGrid at 0x224912b2490>



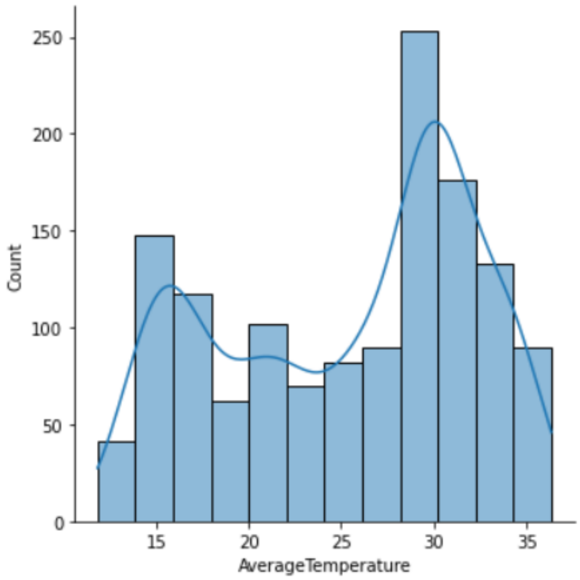

```
delhi = delhi[delhi["dt"] >=  pd.Timestamp('1900-01-01 00:00:00')]
fig , dl = plt.subplots(2,figsize=(15,8))
```

```
dl[0].plot(delhi["dt"] ,delhi["AverageTemperature"])
dl[0].xaxis.set_major_locator(plt.MaxNLocator(12))
dl[0].axhline(y = delhi["AverageTemperature"].mean() , color="red" , linestyle="--")
dl[0].set_title("Average temprature in Delhi")
dl[0].set_xlabel("Date")
dl[0].set_ylabel("Average temprature")
dl[0].grid()

dl[1].plot(delhi["dt"] ,delhi["AverageTemperatureUncertainty"])
dl[1].xaxis.set_major_locator(plt.MaxNLocator(12))
#dl[0].axhline(y = delhi["AverageTemperatureUncertainty"].mean() , color="red" ,
linestyle="--")
dl[1].set_title("Average temprature Uncertainty in Delhi")
dl[1].set_xlabel("Date")
dl[1].set_ylabel("Average temprature Uncertainty")
dl[1].grid()
```




```
sns.displot(delhi["AverageTemperature"] , kde=True)
```

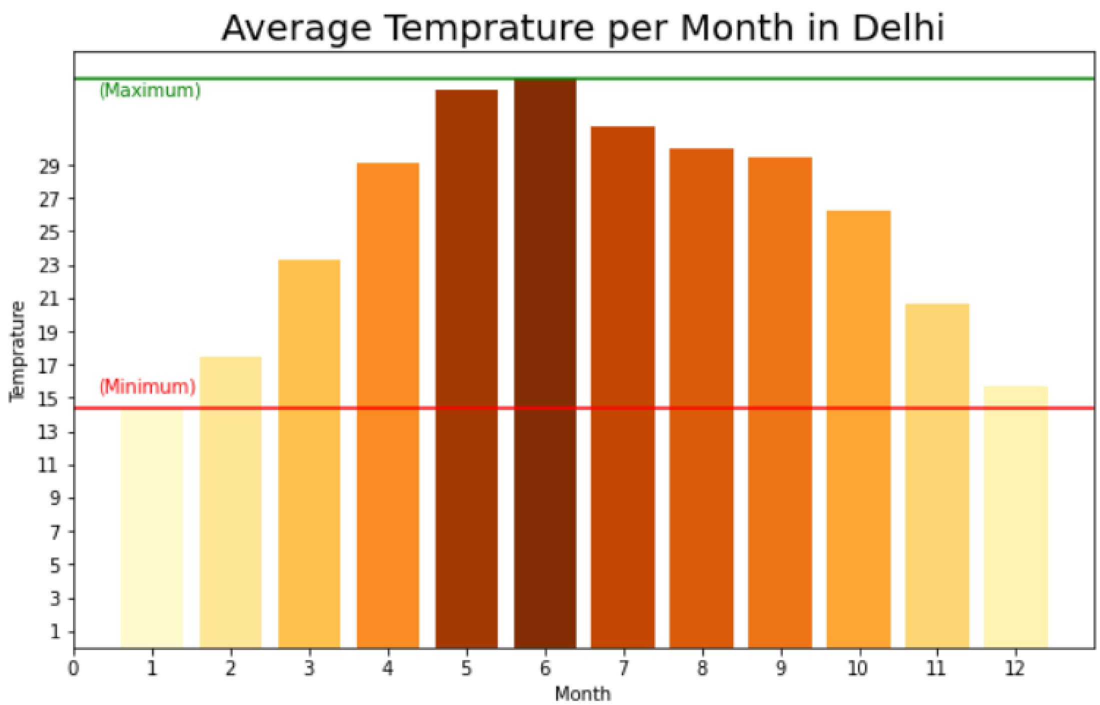
```
<seaborn.axisgrid.FacetGrid at 0x224901643d0>
```




```
temp = delhi.groupby(["Month"]).mean()
temp.drop(columns=["AverageTemperatureUncertainty"] , axis=1 , inplace=True)
temp = temp.sort_values(["AverageTemperature"])
```

```
plt.figure(figsize=(10,6))
plt.bar(temp.index , temp["AverageTemperature"].values ,
color=sns.color_palette("YlOrBr",len(temp.index) ))

plt.axhline(y=temp["AverageTemperature"].values.min() , color="red")
plt.text(.3,temp["AverageTemperature"].values.min()+1 , "(Minimum)" , color='red')
plt.axhline(y=temp["AverageTemperature"].values.max() , color="green")
plt.text(.3,temp["AverageTemperature"].values.max()-1 , "(Maximum)" , color='green')

xticks = plt.xticks(range(13))
yticks = plt.yticks(np.arange(1 , 30 ,2))
plt.xlabel("Month")
plt.ylabel("Temprature")
plt.title("Average Temprature per Month in Delhi" , fontsize=20)
```

```
Text(0.5, 1.0, 'Average Temprature per Month in Delhi')
```




```
surat = india_data[india_data["City"] == "Surat"]
surat.drop(columns=["City" , "Latitude" , "Longitude"] , axis=1 , inplace=True)

#chossing only data after 1900
surat = surat[surat["dt"] >=  pd.Timestamp('1900-01-01 00:00:00')]
surat.head(5)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Month
215200	1900-01-01	20.123	0.921	1
215201	1900-02-01	23.222	0.891	2
215202	1900-03-01	28.264	0.619	3
215203	1900-04-01	30.878	0.750	4
215204	1900-05-01	32.099	0.479	5

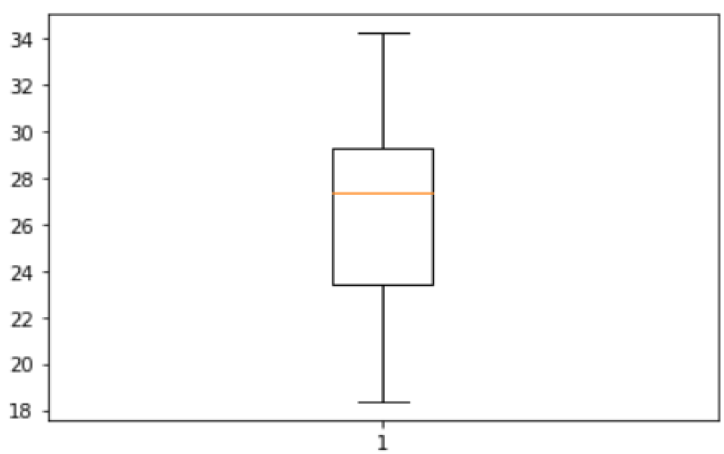
```
surat.tail()
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Month
216559	2013-04-01	30.546	0.279	4
216560	2013-05-01	32.980	1.097	5
216561	2013-06-01	29.418	0.527	6
216562	2013-07-01	27.306	0.257	7
216563	2013-08-01	27.187	0.129	8

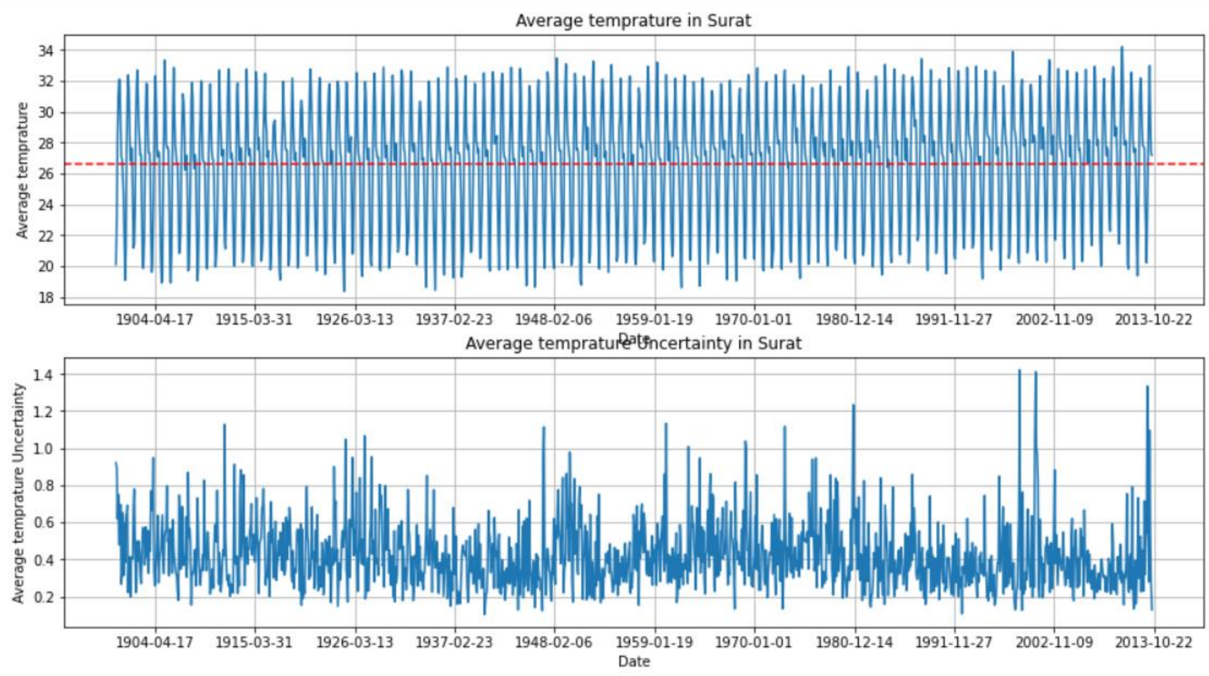
```
surat["AverageTemperature"].describe()
```

```
count    1364.000000
mean       26.605200
std         3.816579
min        18.376000
25%        23.398250
50%        27.306500
75%        29.300000
max        34.211000
Name: AverageTemperature, dtype: float64
```

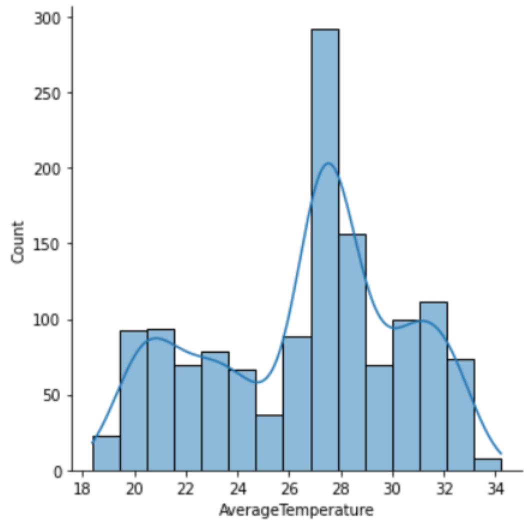
```
fig = plt.boxplot(surat["AverageTemperature"])
```



```
fig , st = plt.subplots(2,figsize=(15,8))
st[0].plot(surat["dt"] ,surat["AverageTemperature"])
st[0].xaxis.set_major_locator(plt.MaxNLocator(12))
st[0].axhline(y = surat["AverageTemperature"].mean() , color="red" , linestyle="--")
st[0].set_title("Average temprature in Surat")
st[0].set_xlabel("Date")
st[0].set_ylabel("Average temprature")
st[0].grid()

st[1].plot(surat["dt"] ,surat["AverageTemperatureUncertainty"])
st[1].xaxis.set_major_locator(plt.MaxNLocator(12))
#st[0].axhline(y = surat["AverageTemperatureUncertainty"].mean() , color="red" ,
linestyle="--")
st[1].set_title("Average temprature Uncertainty in Surat")
st[1].set_xlabel("Date")
st[1].set_ylabel("Average temprature Uncertainty")
st[1].grid()
```



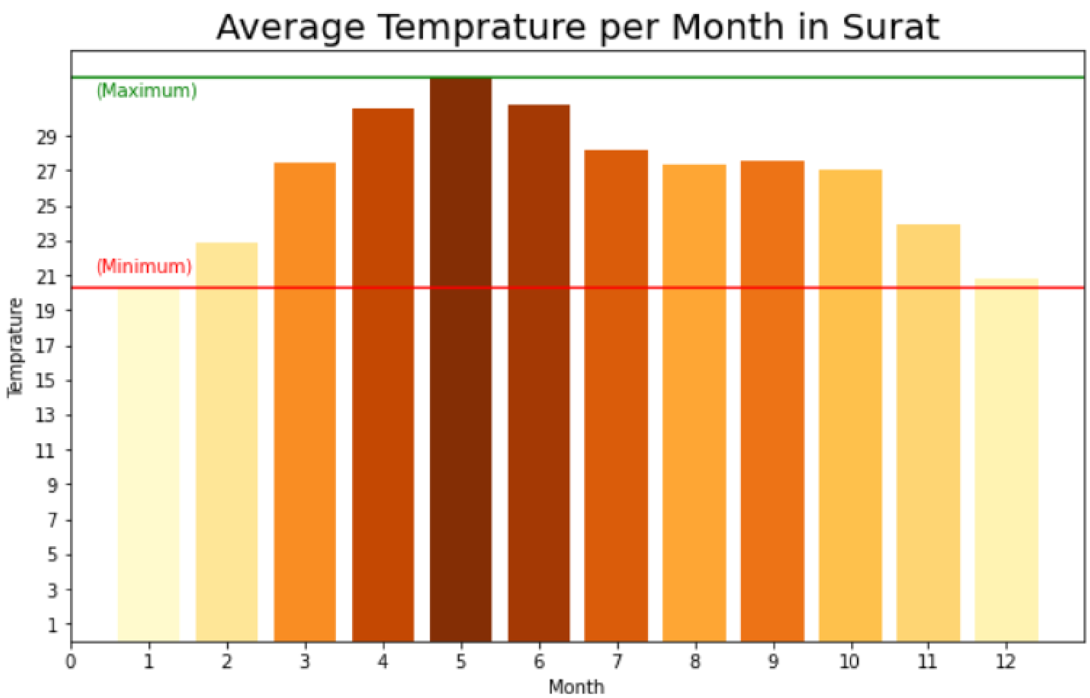
```
<seaborn.axisgrid.FacetGrid at 0x22381a55850>
```




```
temp = surat.groupby(["Month"]).mean()
temp.drop(columns=["AverageTemperatureUncertainty"] , axis=1 , inplace=True)
temp = temp.sort_values(["AverageTemperature"])
```

```
plt.figure(figsize=(10,6))
plt.bar(temp.index , temp["AverageTemperature"].values ,
color=sns.color_palette("YlOrBr",len(temp.index) ))

plt.axhline(y=temp["AverageTemperature"].values.min() , color="red")
plt.text(.3,temp["AverageTemperature"].values.min()+1 , "(Minimum)" , color='red')
plt.axhline(y=temp["AverageTemperature"].values.max() , color="green")
plt.text(.3,temp["AverageTemperature"].values.max()-1 , "(Maximum)" , color='green')

xticks = plt.xticks(range(13))
yticks = plt.yticks(np.arange(1 , 30 ,2))
plt.xlabel("Month")
plt.ylabel("Temprature")
plt.title("Average Temprature per Month in Surat" , fontsize=20)
```

```
Text(0.5, 1.0, 'Average Temprature per Month in Surat')
```



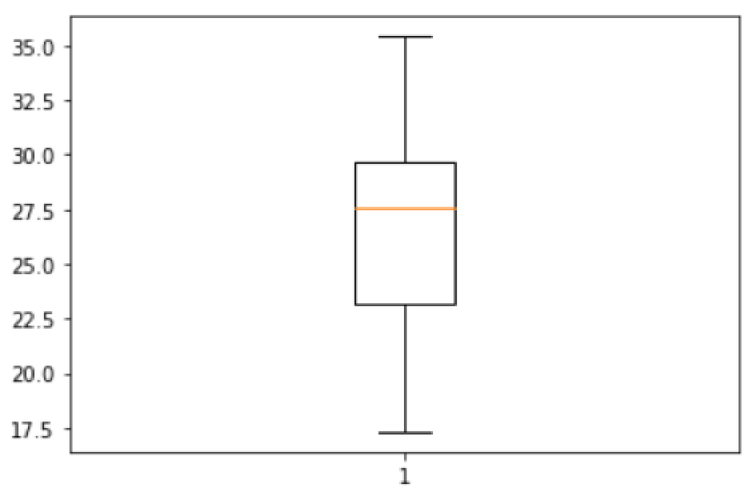

```
ahemdabad = india_data[india_data["City"] == "Ahmadabad"]
ahemdabad.drop(columns=["City" , "Latitude" , "Longitude"] , axis=1 , inplace=True)

#chossing only data after 1900
ahemdabad = ahemdabad[ahemdabad["dt"] >=  pd.Timestamp('1900-01-01 00:00:00')]
ahemdabad.head(5)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Month
5190	1900-01-01	18.814	0.685	1
5191	1900-02-01	22.210	0.720	2
5192	1900-03-01	27.790	0.660	3
5193	1900-04-01	30.873	0.646	4
5194	1900-05-01	32.646	0.557	5

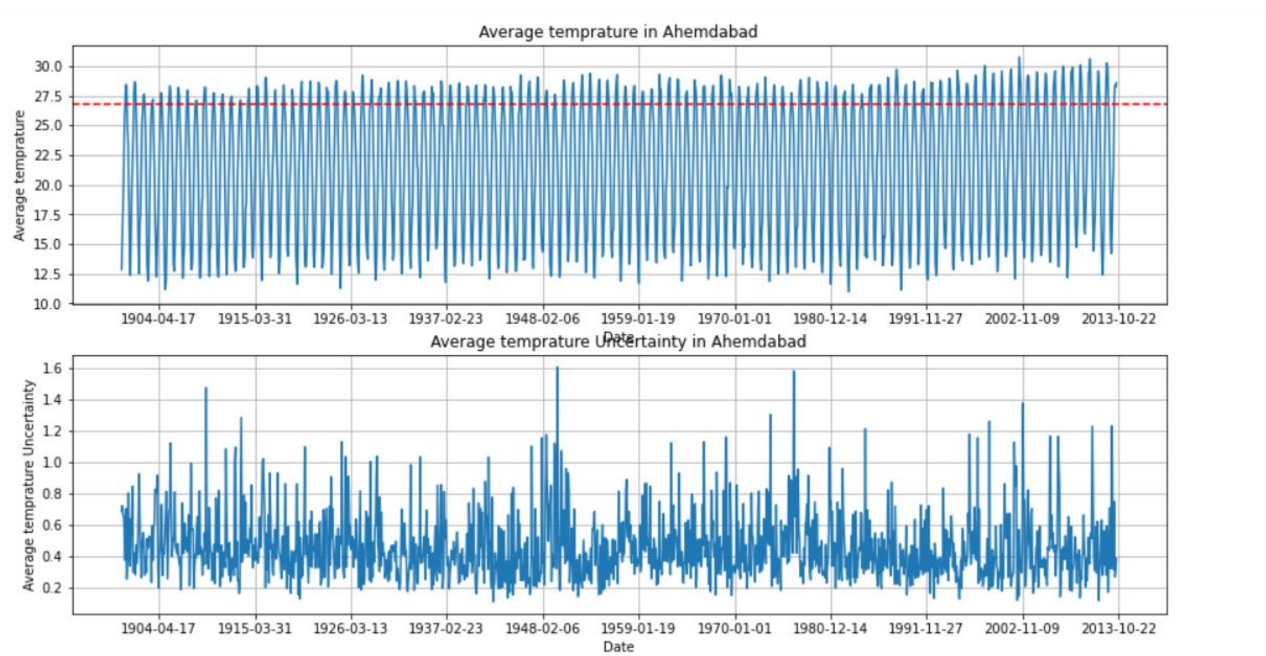
```
ahemdabad["AverageTemperature"].describe()
```

```
count    1364.000000
mean       26.791573
std         4.271054
min        17.320000
25%        23.183750
50%        27.566500
75%        29.688250
max        35.419000
Name: AverageTemperature, dtype: float64
```

fig = plt.boxplot(ahemdabad["AverageTemperature"])

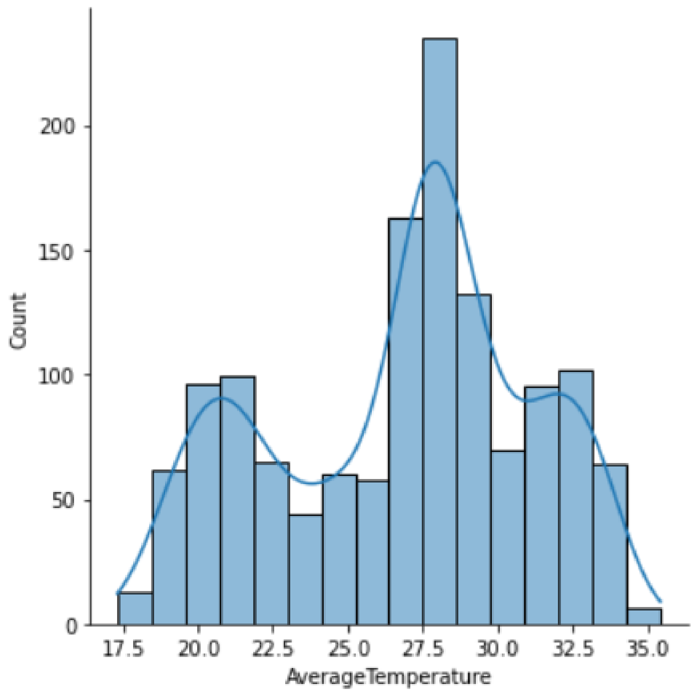


```
fig , ad = plt.subplots(2,figsize=(15,8))
ad[0].plot(ahemdabad["dt"] ,cairo["AverageTemperature"])
ad[0].xaxis.set_major_locator(plt.MaxNLocator(12))
ad[0].axhline(y = ahemdabad["AverageTemperature"].mean() , color="red" , linestyle="--")
ad[0].set_title("Average temprature in Ahemdabad")
ad[0].set_xlabel("Date")
ad[0].set_ylabel("Average temprature")
ad[0].grid()

ad[1].plot(ahemdabad["dt"] ,ahemdabad["AverageTemperatureUncertainty"])
ad[1].xaxis.set_major_locator(plt.MaxNLocator(12))
#ad[0].axhline(y = alex["AverageTemperatureUncertainty"].mean() , color="red" ,
linestyle="--")
ad[1].set_title("Average temprature Uncertainty in Ahemdabad")
ad[1].set_xlabel("Date")
ad[1].set_ylabel("Average temprature Uncertainty")
ad[1].grid()
```



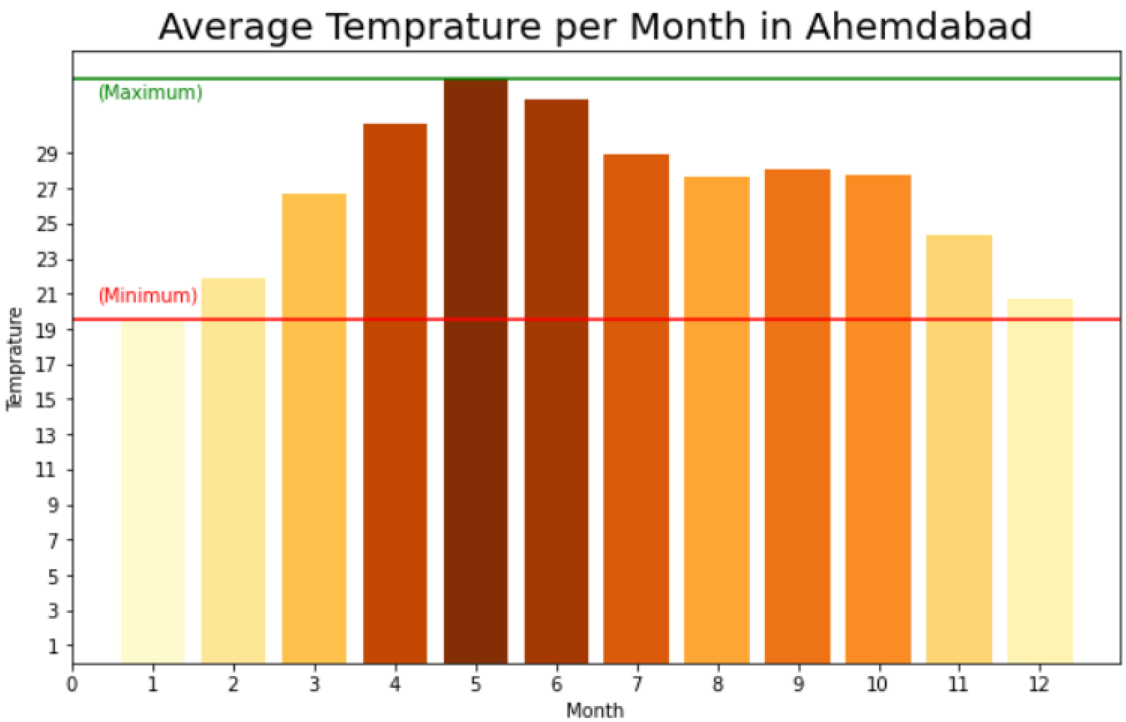

```
sns.displot(ahemdabad["AverageTemperature"] , kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x22380699fd0>
```




```
temp = ahemdabad.groupby(["Month"]).mean()
temp.drop(columns=["AverageTemperatureUncertainty"] , axis=1 , inplace=True)
temp = temp.sort_values(["AverageTemperature"])

plt.figure(figsize=(10,6))
plt.bar(temp.index , temp["AverageTemperature"].values ,
color=sns.color_palette("YlOrBr",len(temp.index) ))

plt.axhline(y=temp["AverageTemperature"].values.min() , color="red")
plt.text(.3,temp["AverageTemperature"].values.min()+1 , "(Minimum)" , color='red')
plt.axhline(y=temp["AverageTemperature"].values.max() , color="green")
plt.text(.3,temp["AverageTemperature"].values.max()-1 , "(Maximum)" , color='green')

xticks = plt.xticks(range(13))
yticks = plt.yticks(np.arange(1 , 30 ,2))
plt.xlabel("Month")
plt.ylabel("Temprature")
plt.title("Average Temprature per Month in Ahemdabad" , fontsize=20)
```

```
Text(0.5, 1.0, 'Average Temprature per Month in Ahemdabad')
```



Conclusion

At the time of writing, the statistical methodology of trend estimation is well elaborated. Some development may come in the form of GLS estimation techniques for nonlinear regression functions, such as the break or the ramp models. Another direction is the development of estimation routines for the piecewise linear model (Fig. 5c). Furthermore, the fitting of multiple change-point models (i.e., more than two change points) is of genuine interest. This is technically challenging and likely necessitates the implementation of advanced optimization techniques, such as genetic algorithms (Michalewicz and Fogel, 2000). The reward of such a technology may consist in a reduction of the problem of fit-interval selection. An interesting example in that regard is the analysis of regional temperatures in the Indian region (Delhi and Surat), which demonstrated accelerated warming in several phases. As regards nonparametric regression, it appears that the potential of that method (standard-error band and derivative estimation) for climatology has only occasionally been appreciated. One example is the search for 14C plateaus (i.e., zero slope) in marine sedimentary records from the Holocene.

The statistical methodology of uncertainty determination for climate time series is well elaborated, as far as uncertainties stemming from measurement or proxy errors in the climate variable, X, are concerned. Bootstrap methods take into account deviations from Gaussian shape. Blocking variants of the bootstrap, such as the MBB, take into account autocorrelation. This means that the two major peculiarities of climate time series—non-Gaussian-shape-and, autocorrelation—can be successfully dealt with in the statistical analysis. As a result, it is possible to avoid unrealistically small error bars from ignored autocorrelation, which could lead to overstatements. On the other hand, as regards timescale errors in the variable T, this is a topic where further research will be quite relevant for paleoclimatology. The book by Mudelsee (2014) contains some algorithms, simulation tests, and references on that emerging field. Also, the Bayesian view of probability can be adopted for the research.