Predicting Impact Of Climate Change By Analyzing Climate Data

Group No: 2

In [1]:

```
# Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
import plotly.express as px
warnings.filterwarnings('ignore')
```

In [2]:

```
# Reading the csv file
df_test = pd.read_csv('conventional_weather_stations_inmet_brazil_1961_2019.csv'

df_test.head(5)
```

Out[2]:

	Estacao	Data	Hora	Precipitacao	TempBulboSeco	TempBulboUmido	TempMaxima	Temp
0	82024	01/01/1961	0	NaN	NaN	NaN	32.3	
1	82024	01/01/1961	1200	NaN	26.0	23.9	NaN	
2	82024	01/01/1961	1800	NaN	32.3	27.0	NaN	
3	82024	02/01/1961	0	NaN	25.8	24.6	33.2	
4	82024	02/01/1961	1200	16.0	26.8	24.0	NaN	

In [3]:

```
# Taking only the necessary Attributes
# #choosing the necessary attributes to perform the required task on it

df = df_test[['Data', 'Precipitacao', 'TempMaxima', 'TempMinima', 'UmidadeRelati
              'DirecaoVento', 'VelocidadeVento', 'Nebulosidade']]
df.head(10)
```

Out[3]:

	Data	Precipitacao	TempMaxima	TempMinima	UmidadeRelativa	DirecaoVento	Velocidade
0	01/01/1961	NaN	32.3	NaN	NaN	NaN	
1	01/01/1961	NaN	NaN	22.9	83.0	5.0	
2	01/01/1961	NaN	NaN	NaN	65.0	5.0	
3	02/01/1961	NaN	33.2	NaN	91.0	9.0	
4	02/01/1961	16.0	NaN	23.7	78.0	5.0	
5	02/01/1961	NaN	NaN	NaN	64.0	5.0	
6	03/01/1961	NaN	32.9	NaN	84.0	5.0	
7	03/01/1961	0.0	NaN	23.3	76.0	9.0	
8	03/01/1961	NaN	NaN	NaN	57.0	5.0	
9	04/01/1961	NaN	30.4	NaN	75.0	5.0	

choosing the necessary attributes to perform the required task on it

In [4]:

```
#Renaming Columns
#renaming the columns such that it is easy for us to understand the data which ar

df.rename(columns = {
      'Data':'Date',
      'Precipitacao':'Precipitation',
      'TempMaxima':'Maximum_Temperature',
      'TempMinima':'Minimum_Temperature',
      'UmidadeRelativa':'Relative_humidity',
      'DirecaoVento':'Wind_direction',
      'VelocidadeVento':'Wind_speed',
      'Nebulosidade':'Cloudiness'
       }, inplace = True)
data = df.copy()
df.head(10)
```

Out[4]:

```
         Date  Precipitation  Maximum_Temperature  Minimum_Temperature  Relative_humidity  Wind_
0  01/01/1961            NaN                 32.3                  NaN                NaN
1  01/01/1961            NaN                  NaN                 22.9               83.0
2  01/01/1961            NaN                  NaN                  NaN               65.0
3  02/01/1961            NaN                 33.2                  NaN               91.0
4  02/01/1961           16.0                  NaN                 23.7               78.0
5  02/01/1961            NaN                  NaN                  NaN               64.0
6  03/01/1961            NaN                 32.9                  NaN               84.0
7  03/01/1961            0.0                  NaN                 23.3               76.0
8  03/01/1961            NaN                  NaN                  NaN               57.0
9  04/01/1961            NaN                 30.4                  NaN               75.0
```

Renaming the columns such that it is easy for us to understand the data which are present in the columns for further work

In [5]:

```
#Understanding information about the data
#presenting the information regarding the data such that required task can be per
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12251335 entries, 0 to 12251334
Data columns (total 8 columns):
 #   Column               Dtype  
---  ------               -----  
 0   Date                 object 
 1   Precipitation        float64
 2   Maximum_Temperature  float64
 3   Minimum_Temperature  float64
 4   Relative_humidity    float64
 5   Wind_direction       float64
 6   Wind_speed           float64
 7   Cloudiness           float64
dtypes: float64(7), object(1)
memory usage: 747.8+ MB
```

presenting the information regarding the data such that required task can be performed on it

In [6]:

```
#Checking for missing values

print(df.isnull().sum())
```

```
Date                         0
Precipitation          8130580
Maximum_Temperature    8300413
Minimum_Temperature    8292199
Relative_humidity       846548
Wind_direction         1221386
Wind_speed             1019529
Cloudiness              433425
dtype: int64
```

checking whether there are missing values to understand about noisy data

In [7]:

```
#checking missing columns in Precipitation

df[df["Precipitation"].isnull()]
```

Out[7]:

```
                Date  Precipitation  Maximum_Temperature  Minimum_Temperature  Relative_humidity
       0  01/01/1961            NaN                 32.3                  NaN                 Na
       1  01/01/1961            NaN                  NaN                 22.9                 83.
       2  01/01/1961            NaN                  NaN                  NaN                 65.
       3  02/01/1961            NaN                 33.2                  NaN                 91.
       5  02/01/1961            NaN                  NaN                  NaN                 64.
     ...         ...            ...                  ...                  ...                  ..
12251328  29/12/2019            NaN                  NaN                  NaN                 43.
12251329  30/12/2019            NaN                 32.1                  NaN                 60.
12251331  30/12/2019            NaN                  NaN                  NaN                 46.
12251332  31/12/2019            NaN                 31.6                  NaN                 55.
12251334  31/12/2019            NaN                  NaN                  NaN                 43.

8130580 rows × 8 columns
```

In [8]:

```
# filling the missing values with their mean
#
meanVal1 = df['Precipitation'].mean()
df['Precipitation'].fillna(value=meanVal1, inplace=True)

meanVal2 = df['Maximum_Temperature'].mean()
df['Maximum_Temperature'].fillna(value=meanVal2, inplace=True)

meanVal3 = df['Minimum_Temperature'].mean()
df['Minimum_Temperature'].fillna(value=meanVal3, inplace=True)

meanVal4 = df['Relative_humidity'].mean()
df['Relative_humidity'].fillna(value=meanVal4, inplace=True)

meanVal5 = df['Wind_direction'].mean()
df['Wind_direction'].fillna(value=meanVal5, inplace=True)

meanVal6 = df['Wind_speed'].mean()
df['Wind_speed'].fillna(value=meanVal6, inplace=True)

meanVal7 = df['Cloudiness'].mean()
df['Cloudiness'].fillna(value=meanVal7, inplace=True)
```

There are certain methods which can be used to replace missing values we here is using the mean method such that the computation is easier

In [9]:

```
# checking for missing values

df.isnull().sum()
```

Out[9]:

```
Date                   0
Precipitation          0
Maximum_Temperature    0
Minimum_Temperature    0
Relative_humidity      0
Wind_direction         0
Wind_speed             0
Cloudiness             0
dtype: int64
```

In [10]:

```
df.head(10)
```

Out[10]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wind_
0	01/01/1961	4.027972	32.300000	18.927569	72.21245	
1	01/01/1961	4.027972	29.725229	22.900000	83.00000	
2	01/01/1961	4.027972	29.725229	18.927569	65.00000	
3	02/01/1961	4.027972	33.200000	18.927569	91.00000	
4	02/01/1961	16.000000	29.725229	23.700000	78.00000	
5	02/01/1961	4.027972	29.725229	18.927569	64.00000	
6	03/01/1961	4.027972	32.900000	18.927569	84.00000	
7	03/01/1961	0.000000	29.725229	23.300000	76.00000	
8	03/01/1961	4.027972	29.725229	18.927569	57.00000	
9	04/01/1961	4.027972	30.400000	18.927569	75.00000	

In [11]:

```
#Checking the number of rows and columns

df.shape
```

Out[11]:

```
(12251335, 8)
```

In [12]:
```
#Printing the last 5 rows

df.tail()
```

Out[12]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity
12251330	30/12/2019	0.000000	29.725229	21.100000	66.0
12251331	30/12/2019	4.027972	29.725229	18.927569	46.0
12251332	31/12/2019	4.027972	31.600000	18.927569	55.0
12251333	31/12/2019	0.000000	29.725229	21.300000	54.0
12251334	31/12/2019	4.027972	29.725229	18.927569	43.0

In [13]:
```
df['Date']=pd.to_datetime(df['Date'])
df.head(5)
```

Out[13]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wind_direc
0	1961-01-01	4.027972	32.300000	18.927569	72.21245	12.7
1	1961-01-01	4.027972	29.725229	22.900000	83.00000	5.0
2	1961-01-01	4.027972	29.725229	18.927569	65.00000	5.0
3	1961-02-01	4.027972	33.200000	18.927569	91.00000	9.0
4	1961-02-01	16.000000	29.725229	23.700000	78.00000	5.0

In [14]:
```
#Descriptive Statistics
#

df.describe()
```

Out[14]:

	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wind_directi
count	1.225134e+07	1.225134e+07	1.225134e+07	1.225134e+07	1.225134e+
mean	4.027972e+00	2.972523e+01	1.892757e+01	7.221245e+01	1.274881e+
std	6.321910e+00	2.592241e+00	2.528762e+00	1.746268e+01	1.053072e+
min	0.000000e+00	-2.000000e+00	-9.000000e+00	8.000000e+00	0.000000e+
25%	1.600000e+00	2.972523e+01	1.892757e+01	6.100000e+01	5.000000e+
50%	4.027972e+00	2.972523e+01	1.892757e+01	7.300000e+01	1.274881e+
75%	4.027972e+00	2.972523e+01	1.892757e+01	8.600000e+01	1.800000e+
max	3.779000e+02	4.470000e+01	3.650000e+01	1.000000e+02	9.900000e+

Describing the required columns

In [15]:

```
#Normalization Using Min Max Scaling
#

x = df.iloc[:,1:7]
global_min = x.min()
global_max = x.max()
df.iloc[:,1:7] = (x-global_min) / (global_max - global_min)


df.describe()
```

Out[15]:

	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wind_directi
count	1.225134e+07	1.225134e+07	1.225134e+07	1.225134e+07	1.225134e+
mean	1.065883e-02	6.793411e-01	6.137927e-01	6.979614e-01	1.287759e-
std	1.672906e-02	5.550837e-02	5.557719e-02	1.898117e-01	1.063709e-
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+
25%	4.233924e-03	6.793411e-01	6.137927e-01	5.760870e-01	5.050505e-
50%	1.065883e-02	6.793411e-01	6.137927e-01	7.065217e-01	1.287759e-
75%	1.065883e-02	6.793411e-01	6.137927e-01	8.478261e-01	1.818182e-
max	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+

The process of transforming the columns in a dataset to the same scale

In [16]:

```
# Getting year,month,day and day name

df['Year']=df["Date"].dt.year
df['Month']=df["Date"].dt.month
df['Day']=df["Date"].dt.day
df['Day Name']=df["Date"].dt.day_name()
df.head(10)
```

Out[16]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wind_directic
0	1961-01-01	0.010659	0.734475	0.613793	0.697961	0.1287
1	1961-01-01	0.010659	0.679341	0.701099	0.815217	0.0505
2	1961-01-01	0.010659	0.679341	0.613793	0.619565	0.0505
3	1961-02-01	0.010659	0.753747	0.613793	0.902174	0.0909
4	1961-02-01	0.042339	0.679341	0.718681	0.760870	0.0505
5	1961-02-01	0.010659	0.679341	0.613793	0.608696	0.0505
6	1961-03-01	0.010659	0.747323	0.613793	0.826087	0.0505
7	1961-03-01	0.000000	0.679341	0.709890	0.739130	0.0909
8	1961-03-01	0.010659	0.679341	0.613793	0.532609	0.0505
9	1961-04-01	0.010659	0.693790	0.613793	0.728261	0.0505

In [17]:

```
# Sorting the data in ascending order on the basis of date

df.sort_values(by="Date",ascending=True,inplace=True)
df.head(10)
```

Out[17]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Win
0	1961-01-01	0.010659	0.734475	0.613793	0.697961	
4227197	1961-01-01	0.010659	0.679341	0.613793	0.293478	
4227196	1961-01-01	0.010659	0.679341	0.727473	0.489130	
4227195	1961-01-01	0.010659	0.824411	0.613793	0.697961	
3206034	1961-01-01	0.010659	0.608137	0.613793	0.697961	
3206035	1961-01-01	0.010659	0.679341	0.624176	0.782609	
3206036	1961-01-01	0.010659	0.679341	0.613793	0.521739	
7519668	1961-01-01	0.010659	0.725910	0.613793	0.697961	
7519669	1961-01-01	0.010659	0.679341	0.595604	0.771739	
7519670	1961-01-01	0.010659	0.679341	0.613793	0.597826	

In [18]:
```
df.tail(10)
```

Out[18]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wi
2421854	2019-12-31	0.010659	0.679341	0.613793	0.456522	
6253606	2019-12-31	0.010659	0.679341	0.613793	0.467391	
6253605	2019-12-31	0.000000	0.679341	0.690110	0.565217	
6253604	2019-12-31	0.010659	0.764454	0.613793	0.673913	
6209680	2019-12-31	0.010659	0.679341	0.613793	0.608696	
6209679	2019-12-31	0.000000	0.679341	0.661538	0.706522	
6209678	2019-12-31	0.010659	0.732334	0.613793	0.673913	
6166564	2019-12-31	0.010659	0.679341	0.613793	0.478261	
12195887	2019-12-31	0.010659	0.679341	0.613793	0.923913	
12251334	2019-12-31	0.010659	0.679341	0.613793	0.380435	

In [19]:
```
df.head(10)
```

Out[19]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Win
0	1961-01-01	0.010659	0.734475	0.613793	0.697961	
4227197	1961-01-01	0.010659	0.679341	0.613793	0.293478	
4227196	1961-01-01	0.010659	0.679341	0.727473	0.489130	
4227195	1961-01-01	0.010659	0.824411	0.613793	0.697961	
3206034	1961-01-01	0.010659	0.608137	0.613793	0.697961	
3206035	1961-01-01	0.010659	0.679341	0.624176	0.782609	
3206036	1961-01-01	0.010659	0.679341	0.613793	0.521739	
7519668	1961-01-01	0.010659	0.725910	0.613793	0.697961	
7519669	1961-01-01	0.010659	0.679341	0.595604	0.771739	
7519670	1961-01-01	0.010659	0.679341	0.613793	0.597826	

In [20]:

```
# Creating a datframe for the year 1961 year records

df_2=df[(df['Year'] == 1961)]
df_2.head(10)
```

Out[20]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Win
0	1961-01-01	0.010659	0.734475	0.613793	0.697961	
4227197	1961-01-01	0.010659	0.679341	0.613793	0.293478	
4227196	1961-01-01	0.010659	0.679341	0.727473	0.489130	
4227195	1961-01-01	0.010659	0.824411	0.613793	0.697961	
3206034	1961-01-01	0.010659	0.608137	0.613793	0.697961	
3206035	1961-01-01	0.010659	0.679341	0.624176	0.782609	
3206036	1961-01-01	0.010659	0.679341	0.613793	0.521739	
7519668	1961-01-01	0.010659	0.725910	0.613793	0.697961	
7519669	1961-01-01	0.010659	0.679341	0.595604	0.771739	
7519670	1961-01-01	0.010659	0.679341	0.613793	0.597826	

In [21]:

```
#  Grouping the records for 1961 dataframe by month

group_1961_month=df_2.groupby('Month')
```

In [22]:

```
# Getting the records for the month of November
df_2[df_2['Month'] == 11]
```

Out[22]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Win
7731764	1961-11-01	0.000000	0.679341	0.509890	0.456522	
6670104	1961-11-01	0.010659	0.679341	0.613793	0.706522	
7731763	1961-11-01	0.010659	0.695931	0.613793	0.554348	
7731765	1961-11-01	0.010659	0.679341	0.613793	0.315217	
6670103	1961-11-01	0.041810	0.679341	0.610989	0.880435	
...	...	...	...	...	...	
3668799	1961-11-30	0.000529	0.679341	0.597802	0.673913	
3668798	1961-11-30	0.010659	0.770878	0.613793	0.717391	
1523885	1961-11-30	0.010659	0.809422	0.613793	0.489130	
3150393	1961-11-30	0.010659	0.679341	0.613793	0.402174	
3206946	1961-11-30	0.010659	0.679341	0.613793	0.391304	

11526 rows × 12 columns

In [23]:

```
# Getting the records for the month of December
df_2[df_2['Month'] == 12]
```

Out[23]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wi
12195922	1961-12-01	0.069331	0.679341	0.567033	0.945652	
12195921	1961-12-01	0.010659	0.556745	0.613793	0.967391	
3316000	1961-12-01	0.010659	0.616702	0.613793	0.869565	
4273221	1961-12-01	0.010659	0.679341	0.613793	0.586957	
9278724	1961-12-01	0.023816	0.679341	0.624176	0.923913	
...	...	...	...	...	...	
8562199	1961-12-31	0.010659	0.679341	0.613793	0.695652	
87079	1961-12-31	0.010659	0.672377	0.613793	0.978261	
7225322	1961-12-31	0.000000	0.679341	0.490110	0.663043	
1476254	1961-12-31	0.000000	0.679341	0.571429	0.815217	
1476253	1961-12-31	0.010659	0.642398	0.613793	0.760870	

11325 rows × 12 columns

In [24]:
```
# Creating a dataframe for the year 2019 year records

df_3=df[(df['Year'] == 2019)]
df_3.head(10)
```

Out[24]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Win
2421013	2019-01-01	0.010659	0.719486	0.613793	0.956522	
2421014	2019-01-01	0.002911	0.679341	0.641758	0.815217	
9182776	2019-01-01	0.010659	0.679341	0.613793	0.293478	
9182775	2019-01-01	0.000000	0.679341	0.716484	0.543478	
8129162	2019-01-01	0.010659	0.728051	0.613793	0.826087	
2987037	2019-01-01	0.010659	0.670236	0.613793	0.836957	
1856439	2019-01-01	0.108759	0.679341	0.685714	0.804348	
1856438	2019-01-01	0.010659	0.721627	0.613793	0.967391	
143531	2019-01-01	0.010659	0.679341	0.613793	0.663043	
143532	2019-01-01	0.000000	0.679341	0.613793	0.945652	

In [25]:
```
#  Grouping the records for 2019 dataframe by month

group_2019_month=df_2.groupby('Month')
```

In [26]:
```
# Getting the records for the month of November 2019

df_3[df_3['Month'] == 11]
```

Out[26]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wi
9624162	2019-11-01	0.010659	0.679341	0.613793	0.532609	
1572690	2019-11-01	0.010659	0.679341	0.613793	0.697961	
10447867	2019-11-01	0.010659	0.679341	0.613793	0.163043	
2473333	2019-11-01	0.010659	0.704497	0.613793	0.782609	
7673930	2019-11-01	0.010659	0.679341	0.613793	0.750000	
...	...	...	...	...	...	
1307171	2019-11-30	0.010659	0.753747	0.613793	0.608696	
10525892	2019-11-30	0.000000	0.679341	0.654945	0.641304	
10525893	2019-11-30	0.010659	0.679341	0.613793	0.315217	
10525891	2019-11-30	0.010659	0.679341	0.613793	0.630435	
10306388	2019-11-30	0.010659	0.679341	0.613793	0.697961	

15256 rows × 12 columns

In [27]:
```
# Desciptive statistics for November 1961

df_2[df_2['Month'] == 11].describe()
```

Out[27]:

	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wind_directi
count	11526.000000	11526.000000	11526.000000	11526.000000	11526.0000
mean	0.010268	0.681382	0.611339	0.679285	0.1163
std	0.014870	0.052272	0.047805	0.187579	0.1111
min	0.000000	0.226981	0.215385	0.065217	0.0000
25%	0.004234	0.679341	0.613793	0.565217	0.0000
50%	0.010659	0.679341	0.613793	0.697961	0.0909
75%	0.010659	0.679341	0.613793	0.826087	0.1818
max	0.339772	0.905782	0.795604	1.000000	0.3636

In [28]:
```
# Desciptive statistics for November 2005

df_3[df_3['Month'] == 11].describe()
```

Out[28]:

	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Wind_directi
count	15256.000000	15256.000000	15256.000000	15256.000000	15256.0000
mean	0.010914	0.693431	0.628259	0.680992	0.1255
std	0.019225	0.051442	0.049350	0.190286	0.1026
min	0.000000	0.370450	0.237363	0.054348	0.0000
25%	0.001058	0.679341	0.613793	0.554348	0.0505
50%	0.010659	0.679341	0.613793	0.697961	0.1287
75%	0.010659	0.679341	0.619780	0.826087	0.1414
max	0.464144	0.942184	0.848352	1.000000	0.3636

In [29]:
```
# Desciptive Satistics for all the month for the year 1961

df_2.groupby('Month').describe()
```

Out[29]:

	Precipitation								Maximum_Tempe	
	count	mean	std	min	25%	50%	75%	max	count	mean
Month										
1	12028.0	0.012500	0.019973	0.0	0.010659	0.010659	0.010659	0.361736	12028.0	0.6
2	10795.0	0.012589	0.021801	0.0	0.010659	0.010659	0.010659	0.429214	10795.0	0.6
3	11936.0	0.011657	0.019052	0.0	0.010659	0.010659	0.010659	0.358296	11936.0	0.6
4	11643.0	0.011157	0.018138	0.0	0.007674	0.010659	0.010659	0.420217	11643.0	0.6
5	11964.0	0.010214	0.015903	0.0	0.002646	0.010659	0.010659	0.351680	11964.0	0.6
6	11606.0	0.010086	0.014774	0.0	0.003771	0.010659	0.010659	0.288436	11606.0	0.6
7	11458.0	0.009441	0.012018	0.0	0.001852	0.010659	0.010659	0.237894	11458.0	0.6
8	11557.0	0.009181	0.011527	0.0	0.000529	0.010659	0.010659	0.216459	11557.0	0.6
9	11668.0	0.009620	0.013894	0.0	0.000265	0.010659	0.010659	0.313840	11668.0	0.6
10	12026.0	0.010561	0.016263	0.0	0.005557	0.010659	0.010659	0.354856	12026.0	0.6
11	11526.0	0.010268	0.014870	0.0	0.004234	0.010659	0.010659	0.339772	11526.0	0.6
12	11325.0	0.011476	0.017250	0.0	0.010585	0.010659	0.010659	0.341360	11325.0	0.6

12 rows × 72 columns

In [30]:

```
# Desciptive Satistics for all the month for the year 2019

df_3.groupby('Month').describe()
```

Out[30]:

	Precipitation								Maximum_Tempe	
	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**	**count**	**mean**
Month										
1	16023.0	0.010881	0.016829	0.0	0.002382	0.010659	0.010659	0.429743	16023.0	0.6
2	14964.0	0.012337	0.020475	0.0	0.008203	0.010659	0.010659	0.343212	14964.0	0.6
3	16247.0	0.012226	0.021521	0.0	0.007145	0.010659	0.010659	0.620270	16247.0	0.6
4	15811.0	0.011087	0.017632	0.0	0.003175	0.010659	0.010659	0.375496	15811.0	0.6
5	16350.0	0.010698	0.017785	0.0	0.001588	0.010659	0.010659	0.525536	16350.0	0.6
6	15249.0	0.009633	0.014349	0.0	0.000265	0.010659	0.010659	0.482667	15249.0	0.6
7	15170.0	0.009709	0.014335	0.0	0.000000	0.010659	0.010659	0.347976	15170.0	0.6
8	16013.0	0.008976	0.012982	0.0	0.000000	0.010659	0.010659	0.451442	16013.0	0.6
9	15131.0	0.009199	0.013176	0.0	0.000000	0.010659	0.010659	0.378936	15131.0	0.6
10	15642.0	0.010115	0.015872	0.0	0.000000	0.010659	0.010659	0.379995	15642.0	0.6
11	15256.0	0.010914	0.019225	0.0	0.001058	0.010659	0.010659	0.464144	15256.0	0.6
12	15213.0	0.010389	0.016873	0.0	0.000529	0.010659	0.010659	0.311723	15213.0	0.6

12 rows × 72 columns

In [31]:

```
# Getting mean,std deviation and median for all the months in the year 1961

df_2.groupby('Month').agg(['mean','std','median'])
```

Out[31]:

	Precipitation			Maximum_Temperature			Minimum_Temperature		
	mean	**std**	**median**	**mean**	**std**	**median**	**mean**	**std**	**median**
Month									
1	0.012500	0.019973	0.010659	0.676707	0.045991	0.679341	0.613500	0.048226	0.613793
2	0.012589	0.021801	0.010659	0.675353	0.046444	0.679341	0.612712	0.044793	0.613793
3	0.011657	0.019052	0.010659	0.675365	0.048482	0.679341	0.609511	0.049326	0.613793
4	0.011157	0.018138	0.010659	0.670633	0.054232	0.679341	0.605623	0.055709	0.613793
5	0.010214	0.015903	0.010659	0.665902	0.057372	0.679341	0.597066	0.066128	0.613793
6	0.010086	0.014774	0.010659	0.658584	0.074512	0.679341	0.591006	0.074992	0.613793
7	0.009441	0.012018	0.010659	0.663706	0.063810	0.679341	0.589542	0.074648	0.613793
8	0.009181	0.011527	0.010659	0.673364	0.057476	0.679341	0.597310	0.062103	0.613793
9	0.009620	0.013894	0.010659	0.679690	0.062444	0.679341	0.604803	0.054289	0.613793
10	0.010561	0.016263	0.010659	0.674995	0.055738	0.679341	0.609706	0.049446	0.613793
11	0.010268	0.014870	0.010659	0.681382	0.052272	0.679341	0.611339	0.047805	0.613793
12	0.011476	0.017250	0.010659	0.676231	0.049950	0.679341	0.612787	0.046035	0.613793

12 rows × 27 columns

In [32]:

```
# Getting mean,std deviation and median for all the months in the year 2019

df_3.groupby('Month').agg(['mean','std','median'])
```

Out[32]:

	Precipitation			Maximum_Temperature			Minimum_Temperature		
	mean	std	median	mean	std	median	mean	std	median
Month									
1	0.010881	0.016829	0.010659	0.696400	0.047254	0.679341	0.629658	0.044716	0.613793
2	0.012337	0.020475	0.010659	0.691441	0.047328	0.679341	0.629582	0.047836	0.613793
3	0.012226	0.021521	0.010659	0.689203	0.048305	0.679341	0.628573	0.048885	0.613793
4	0.011087	0.017632	0.010659	0.688245	0.048775	0.679341	0.626915	0.049804	0.613793
5	0.010698	0.017785	0.010659	0.682439	0.056432	0.679341	0.620809	0.053980	0.613793
6	0.009633	0.014349	0.010659	0.681463	0.052252	0.679341	0.614618	0.056774	0.613793
7	0.009709	0.014335	0.010659	0.678874	0.058404	0.679341	0.607955	0.063788	0.613793
8	0.008976	0.012982	0.010659	0.684877	0.060634	0.679341	0.612358	0.061553	0.613793
9	0.009199	0.013176	0.010659	0.693958	0.062611	0.679341	0.621193	0.054348	0.613793
10	0.010115	0.015872	0.010659	0.694739	0.055981	0.679341	0.625494	0.049311	0.613793
11	0.010914	0.019225	0.010659	0.693431	0.051442	0.679341	0.628259	0.049350	0.613793
12	0.010389	0.016873	0.010659	0.696196	0.050021	0.679341	0.630388	0.048678	0.613793

12 rows × 27 columns

In [33]:

```
# Getting mean,sd deviation and median values for the winter season of 1961

df_2[(df_2['Month'] == 12) | (df_2['Month'] == 1) | (df_2['Month'] == 2)].groupby
```

Out[33]:

	Precipitation			Maximum_Temperature			Minimum_Temperature		
	mean	std	median	mean	std	median	mean	std	median
Month									
1	0.012500	0.019973	0.010659	0.676707	0.045991	0.679341	0.613500	0.048226	0.613793
2	0.012589	0.021801	0.010659	0.675353	0.046444	0.679341	0.612712	0.044793	0.613793
12	0.011476	0.017250	0.010659	0.676231	0.049950	0.679341	0.612787	0.046035	0.613793

3 rows × 27 columns

In [34]:

```
# Grouping the dataframe for the year 1961 about 'Month' and 'Day Name ' columns

df_2.groupby(['Month','Day Name']).count()
```

Out[34]:

		Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_hun
Month	**Day Name**					
1	**Friday**	1560	1560	1560	1560	
	Monday	1940	1940	1940	1940	
	Saturday	1547	1547	1547	1547	
	Sunday	1919	1919	1919	1919	
	Thursday	1553	1553	1553	1553	
...	**...**	...	...	...	...	
12	**Saturday**	1842	1842	1842	1842	
	Sunday	1846	1846	1846	1846	
	Thursday	1428	1428	1428	1428	
	Tuesday	1450	1450	1450	1450	
	Wednesday	1454	1454	1454	1454	

84 rows × 10 columns

In [35]:

```
# Getting the count of each day foe each month in 1961

df_2.groupby(['Month','Day Name']).count()
```

Out[35]:

		Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_hun
Month	**Day Name**					
1	**Friday**	1560	1560	1560	1560	
	Monday	1940	1940	1940	1940	
	Saturday	1547	1547	1547	1547	
	Sunday	1919	1919	1919	1919	
	Thursday	1553	1553	1553	1553	
...	**...**	...	...	...	...	
12	**Saturday**	1842	1842	1842	1842	
	Sunday	1846	1846	1846	1846	
	Thursday	1428	1428	1428	1428	
	Tuesday	1450	1450	1450	1450	
	Wednesday	1454	1454	1454	1454	

84 rows × 10 columns

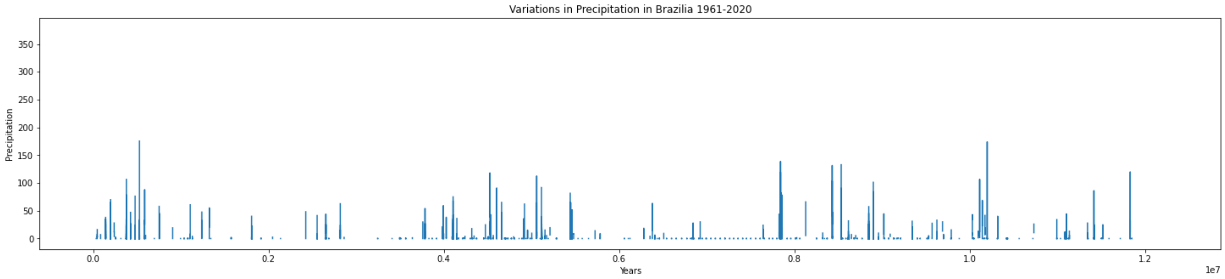
In [36]:
```
df.head(10)
```

Out[36]:

	Date	Precipitation	Maximum_Temperature	Minimum_Temperature	Relative_humidity	Win
0	1961-01-01	0.010659	0.734475	0.613793	0.697961	
4227197	1961-01-01	0.010659	0.679341	0.613793	0.293478	
4227196	1961-01-01	0.010659	0.679341	0.727473	0.489130	
4227195	1961-01-01	0.010659	0.824411	0.613793	0.697961	
3206034	1961-01-01	0.010659	0.608137	0.613793	0.697961	
3206035	1961-01-01	0.010659	0.679341	0.624176	0.782609	
3206036	1961-01-01	0.010659	0.679341	0.613793	0.521739	
7519668	1961-01-01	0.010659	0.725910	0.613793	0.697961	
7519669	1961-01-01	0.010659	0.679341	0.595604	0.771739	
7519670	1961-01-01	0.010659	0.679341	0.613793	0.597826	

Data Visualization

In [37]:
```
# Plotting the variation in precipitation in Brazil between the years 1961 to 201
#
plt.figure(figsize = (25,5));
plt.plot(data.Precipitation);
plt.ylabel('Precipitation');
plt.xlabel('Years')
plt.title('Variations in Precipitation in Brazilia 1961-2020');
```



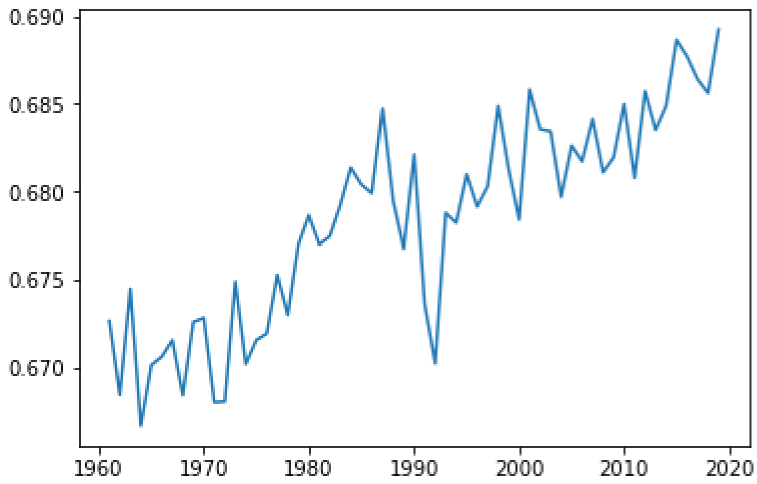
From the graph there is varience in the precipitaion level in brazil from 1961 to 2019. There is a certain rise and fall of the level.

In [38]:

```
# Plotting the yearly maximum temperature and the year

df.index = pd.to_datetime(df['Date'])
average_monthly_temperature = df['Maximum_Temperature'].groupby(df.index.year).me

x = np.arange(1961, 2020)

l = average_monthly_temperature.tolist()

average_monthly_temperature
plt.plot(x, l, label="line L")
```

Out[38]:

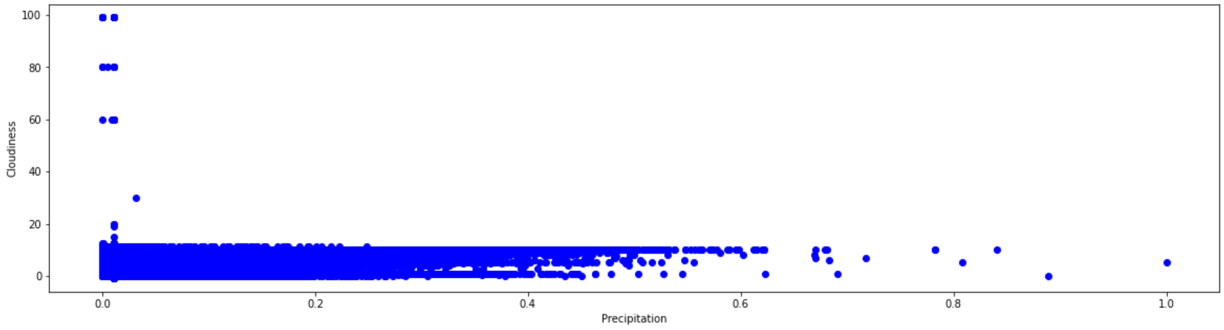
```
[<matplotlib.lines.Line2D at 0x20d8587f5e0>]
```



The above graph shows us the variance of maximum temperature to that of the year and we can see the there is a hugh rise in temperature in 2020 and there is downfall in between 1990 to 2000.

In [39]:

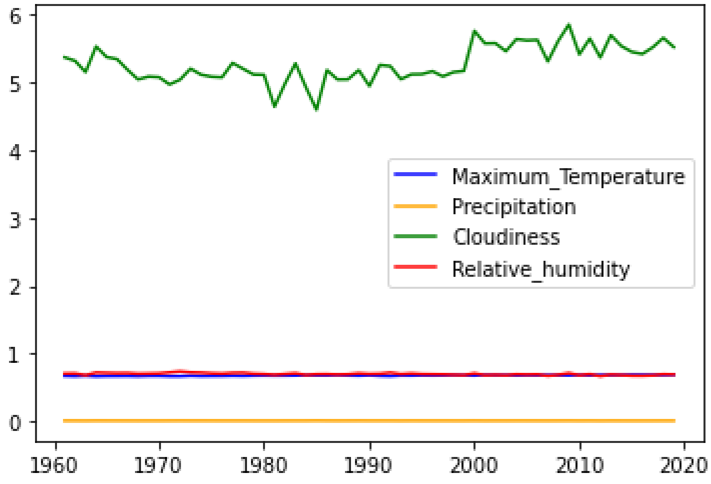
```
# Drawing a scatterplot between precipitation and cloudiness

col1 = df['Precipitation']
col2 = df['Cloudiness']

plt.figure(figsize = (20, 5))
plt.scatter(col1, col2,color="blue",)
plt.xlabel('Precipitation')
plt.ylabel('Cloudiness')
plt.show()
```



The above graph shows us how precipitaion is affected by cloudiness

In [40]:
```
# Plotting the relationship between cloudiness,temperature,precipitation and rela

average_monthly_temperature = df['Maximum_Temperature'].groupby(df.index.year).me
average_monthly_precipitation = df['Precipitation'].groupby(df.index.year).mean(
average_monthly_cloud = df['Cloudiness'].groupby(df.index.year).mean()
average_monthly_humidity = df['Relative_humidity'].groupby(df.index.year).mean()

a = average_monthly_temperature.tolist()
b = average_monthly_precipitation.tolist()
c = average_monthly_cloud.tolist()
d = average_monthly_humidity.tolist()

plt.plot(x, a, label="Maximum_Temperature",color="blue")
plt.plot(x, b, label="Precipitation",color="orange")
plt.plot(x, c, label="Cloudiness",color="green")
plt.plot(x, d, label="Relative_humidity",color="red")
plt.legend()
```

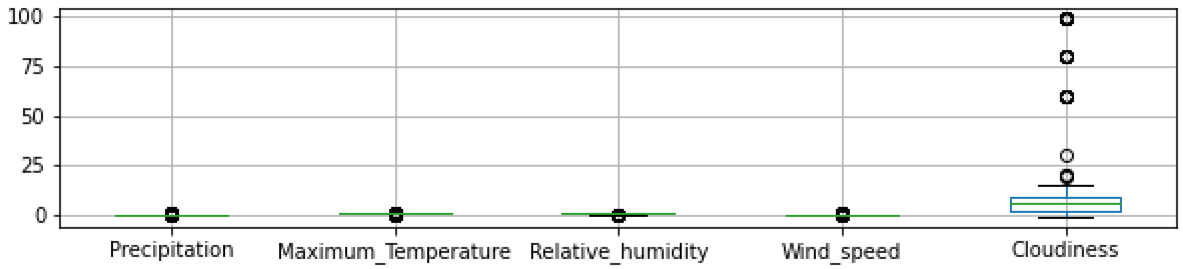
Out[40]:
```
<matplotlib.legend.Legend at 0x20d21467970>
```



The above graph describes that precipitation, maximum temperature and relative humidity is almost constant but the cloudiness varies so much throughout

```
In [41]: # Boxplot

         df.boxplot(column=['Precipitation','Maximum_Temperature','Relative_humidity','Wir
```

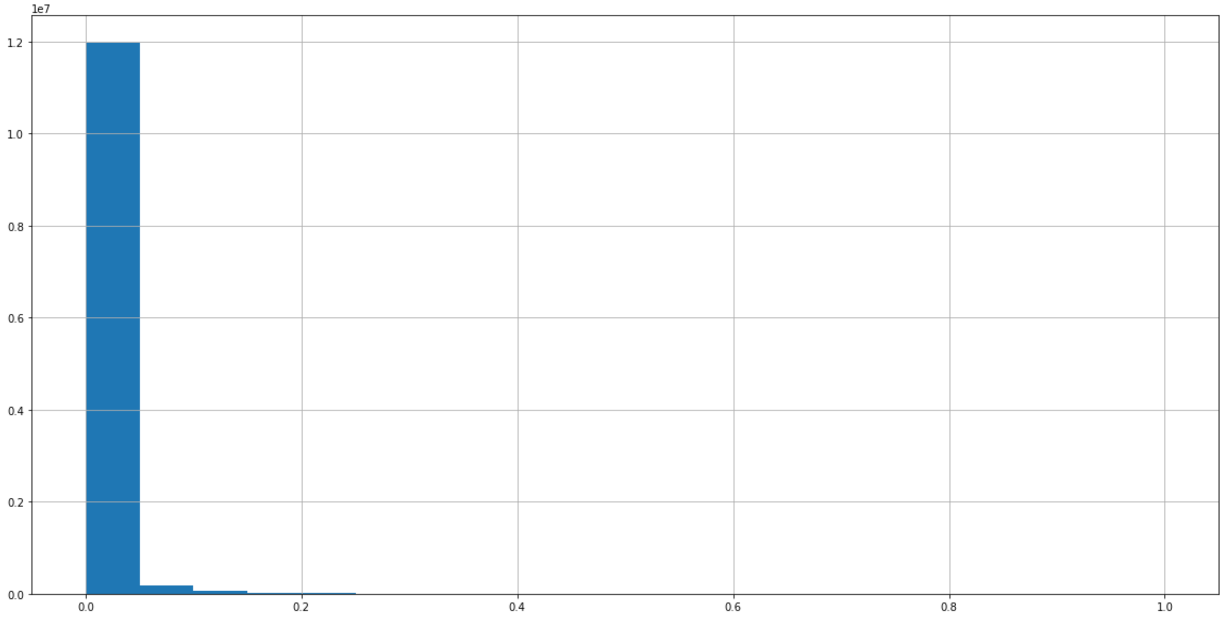
```
Out[41]: <AxesSubplot:>
```



The box plot shows the distribution of the attributes it is being compared with one another.

```
In [42]: # Plotting the histogram for the precipitation

         df['Precipitation'].hist(bins=20,figsize=(20,10))
```

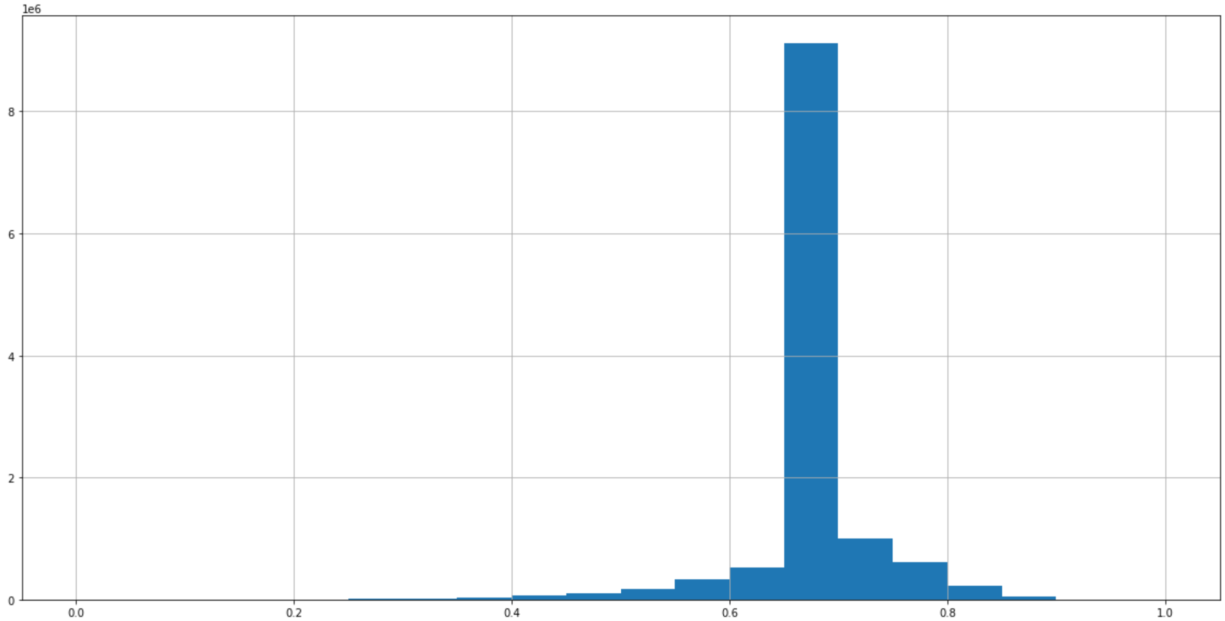
```
Out[42]: <AxesSubplot:>
```



The graph displays the continues values which explains the rise and then fall of precipitation

In [43]:
```
# Plotting the histogram for the maximum temperature

df['Maximum_Temperature'].hist(bins=20,figsize=(20,10))
```

Out[43]:
```
<AxesSubplot:>
```

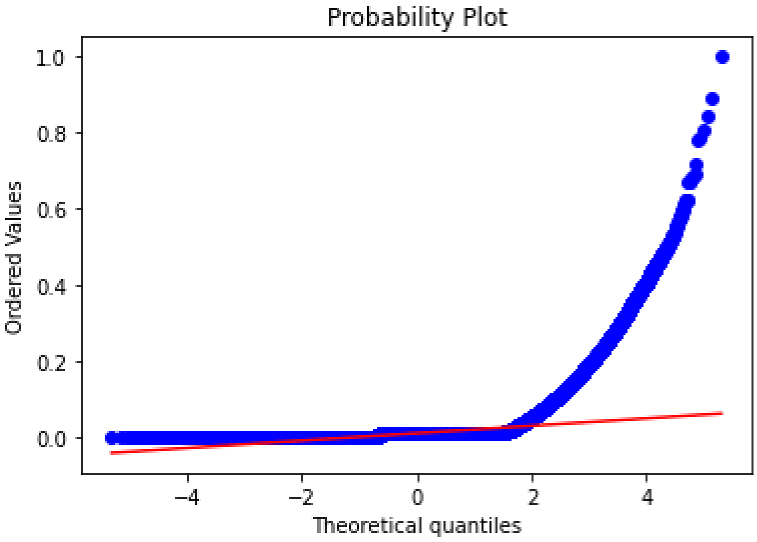


The above graph shows the continues values from low to a certain rise and going low of the attibute Maximum_temperature

In [44]:
```
# Create twodimensional NumPy arrays from onedimensional Pandas series for the fe

X = df[['Precipitation','Maximum_Temperature','Relative_humidity','Wind_speed','C
y = df['Year'].values
print(X)
print(y)
```

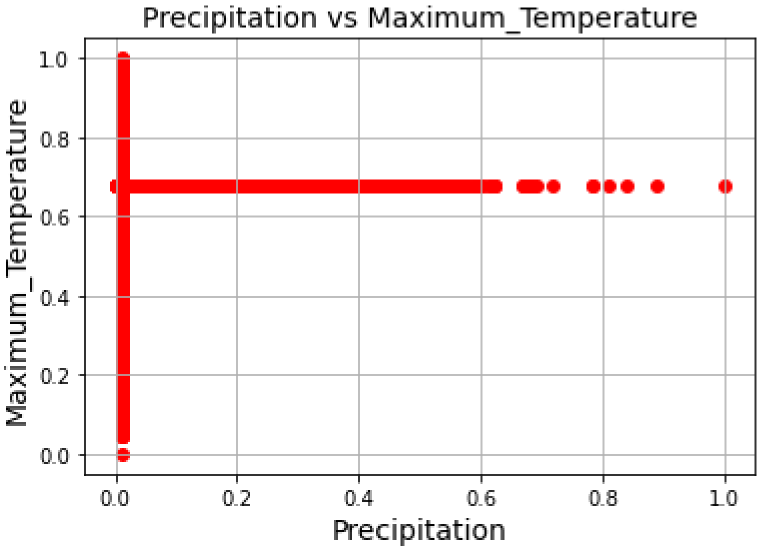
```
[[ 0.01065883  0.73447537  0.69796141  0.08104219  5.30951457]
 [ 0.01065883  0.67934109  0.29347826  0.04135875  5.        ]
 [ 0.01065883  0.67934109  0.48913043  0.0827175   4.        ]
 ...
 [ 0.01065883  0.67934109  0.47826087  0.08510638 10.        ]
 [ 0.01065883  0.67934109  0.92391304  0.04255319 10.        ]
 [ 0.01065883  0.67934109  0.38043478  0.          8.75      ]]
[1961 1961 1961 ... 2019 2019 2019]
```

In [45]:
```
# Apply the 'norm.pdf()' function to get the probabilities and then create a norm

import scipy.stats as stats
import pylab

stats.probplot(df['Precipitation'], dist="norm", plot=pylab)
pylab.show()
```



This graph displays the probabilities of theoretical quantiles to that of oredered values. Here percipitaion is having a rise in (2,1.0)

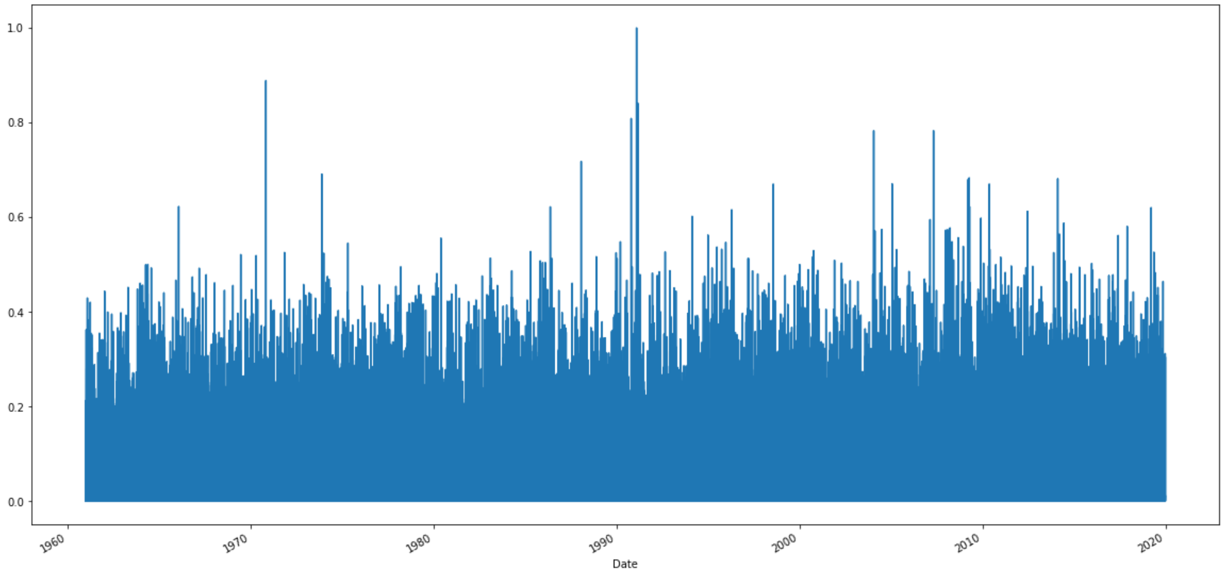
In [46]:
```
# Create a scatter plot between the errors and the independent variable for the t

plt.scatter(df['Precipitation'], df['Maximum_Temperature'], color = 'red')
plt.title('Precipitation vs Maximum_Temperature', fontsize = 14)
plt.xlabel('Precipitation', fontsize = 14)
plt.ylabel('Maximum_Temperature', fontsize = 14)
plt.grid(True)
plt.show()
```



The graph between Maximum_temperature to that of percipitation.Here the graph displays how much Temperature is affected with percipitation

In [47]:
```
#Time-Series Line Plots

df['Precipitation'].plot(figsize=(20,10))
```

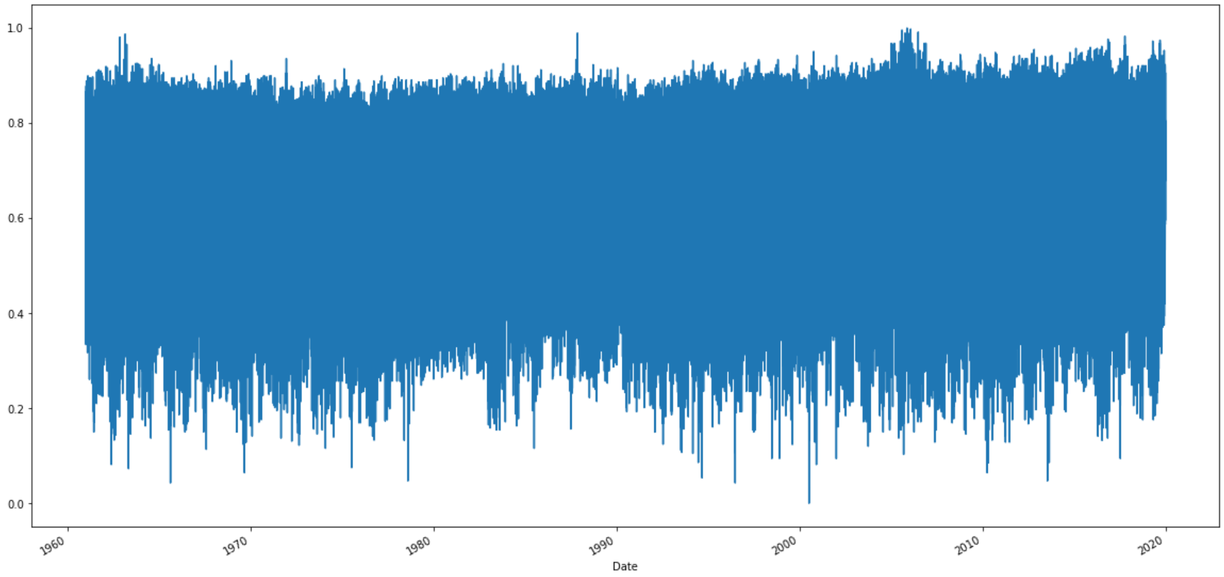
Out[47]:
```
<AxesSubplot:xlabel='Date'>
```



This graph is a time graph it tells us that during 1990 the percipitation is the highest recorded according to the data.It is a very much scattered garph

In [48]:
```
# Time-Series Line Plots

df['Maximum_Temperature'].plot(figsize=(20,10))
```

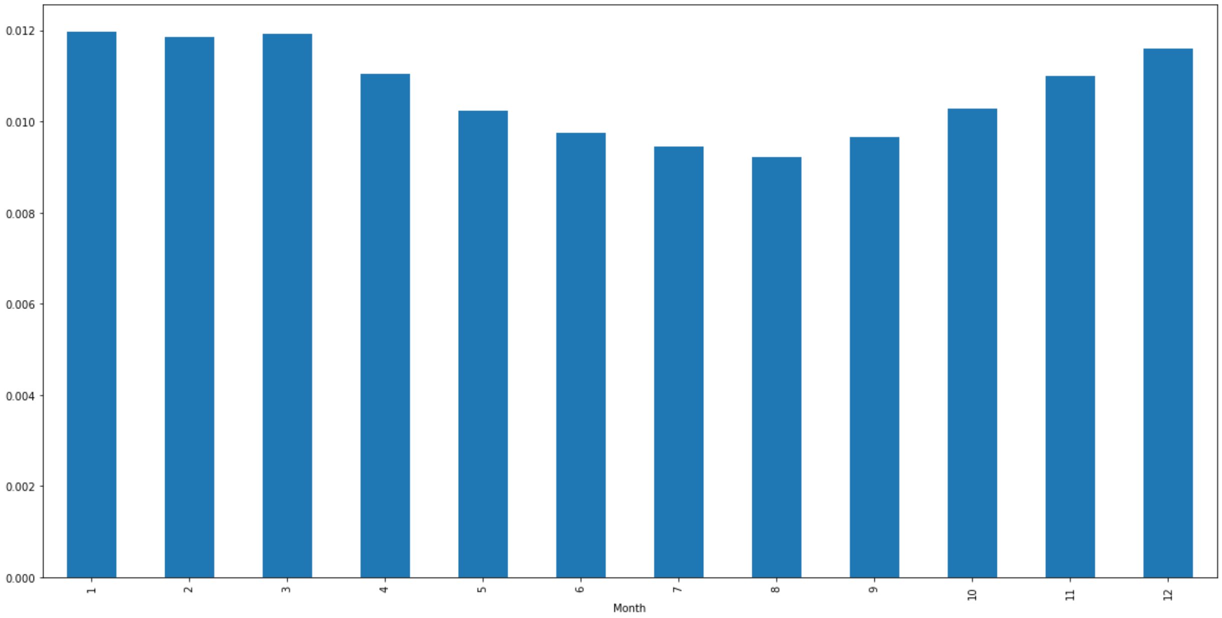
Out[48]:
```
<AxesSubplot:xlabel='Date'>
```



This graph is a time graph it tells us that during 2000 the Maximum_Temperature is the least recorded according to the data.It is a scattered graph

```
In [49]: # Bivariate Bar Plots

         df.groupby('Month')['Precipitation'].mean().plot(kind='bar',figsize=(20,10))
```

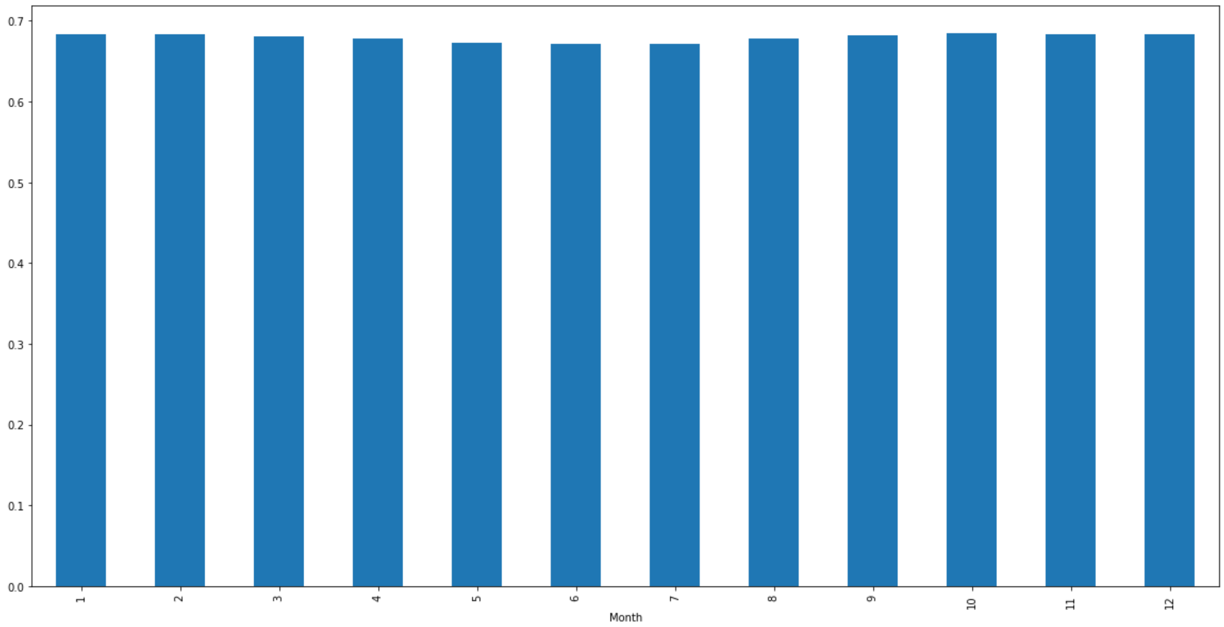
```
Out[49]: <AxesSubplot:xlabel='Month'>
```



The graph showing is bar graph and it tells us that in 3rd month the recorded precipitation is the highest.

In [50]:
```
# Bivariate Bar Plots

df.groupby('Month')['Maximum_Temperature'].mean().plot(kind='bar',figsize=(20,10)
```

Out[50]:
```
<AxesSubplot:xlabel='Month'>
```



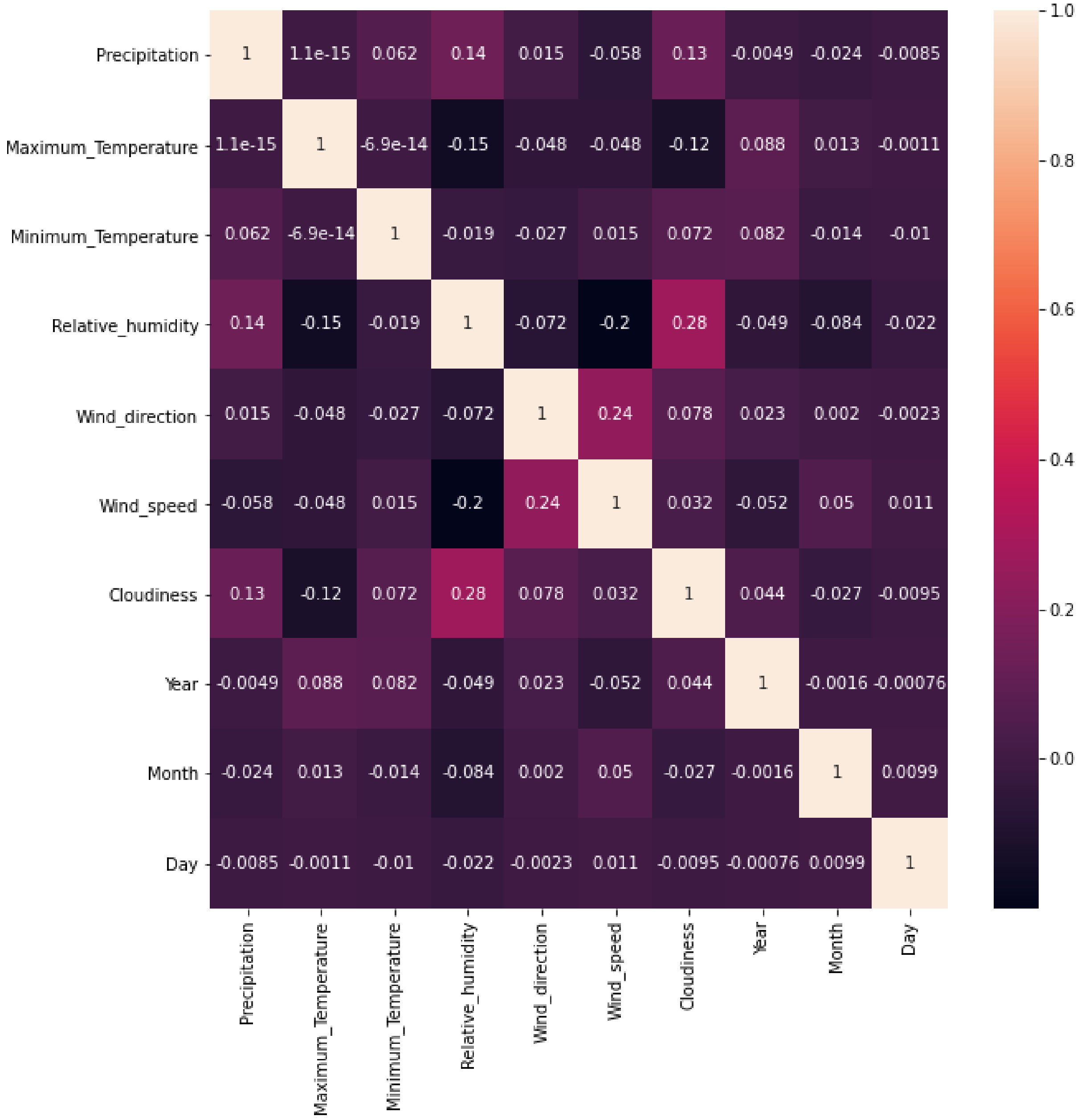
The graph showing is bar graph and it tells us that in 12th month the recorded Maximum_Temperature is the highest.

In [51]:

```
# Correlation
import seaborn as sns
df.corr()

# Heatmap

plt.figure(figsize=(10,10))

sns.heatmap(df.corr(),annot=True)
```

Out[51]:

```
<AxesSubplot:>
```



Splitting the dataset into training and testing

In [52]:
```
import datetime as dt
from sklearn.model_selection import train_test_split

year = pd.to_datetime(df.Date).dt.year
train_df = df[year == 2009]
val_df = df[year == 2013]
test_df = df[year == 2011]

train_df= train_df.drop(['Date'], axis=1)
val_df= val_df.drop(['Date'], axis=1)
test_df= test_df.drop(['Date'], axis=1)

train_df.info()
print(len(train_df), len(val_df), len(test_df))
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 274980 entries, 2009-01-01 to 2009-12-31
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   Precipitation        274980 non-null  float64
 1   Maximum_Temperature  274980 non-null  float64
 2   Minimum_Temperature  274980 non-null  float64
 3   Relative_humidity    274980 non-null  float64
 4   Wind_direction       274980 non-null  float64
 5   Wind_speed           274980 non-null  float64
 6   Cloudiness           274980 non-null  float64
 7   Year                 274980 non-null  int64  
 8   Month                274980 non-null  int64  
 9   Day                  274980 non-null  int64  
 10  Day Name             274980 non-null  object 
dtypes: float64(7), int64(3), object(1)
memory usage: 25.2+ MB
274980 265870 274561
```

Making Target column

Making RainOrNot as the target column and all other column as input columns

In [53]:
```
def f(row):
  if row['Precipitation'] > 0:
    val = 1
  else:
    val = 0
  return val
```

In [54]:
```
train_df['RainOrNot'] = train_df.apply(f, axis = 1)
test_df['RainOrNot'] = test_df['Precipitation'].map(lambda x: 1 if x > 0.0 else
val_df['RainOrNot'] = val_df['Precipitation'].map(lambda x: 1 if x > 0.0 else 0)
```

In [73]:

```
input_cols = list(train_df.columns)[1:7]
target_col = 'RainOrNot'
print('Input Features: ', input_cols)
print()
print('Target Feature: ', target_col)
```

```
Input Features:  ['Maximum_Temperature', 'Minimum_Temperature', 'Relative_humid
ity', 'Wind_direction', 'Wind_speed', 'Cloudiness']

Target Feature:  RainOrNot
```

In [95]:

```
train_inputs = train_df[input_cols].copy()
train_targets = train_df[target_col].copy()

val_inputs = val_df[input_cols].copy()
val_targets = val_df[target_col].copy()

test_inputs = test_df[input_cols].copy()
test_targets = test_df[target_col].copy()
```

Implementing Logistic Regression

In [99]:

```

from sklearn.linear_model import LogisticRegression
scores_dict={}
model = LogisticRegression(solver='liblinear')
model.fit(train_inputs, train_targets)

train_preds = model.predict(train_inputs)
train_probs = model.predict_proba(train_inputs)
```

In [100]:

```
from sklearn.metrics import accuracy_score
accuracy_score(train_targets, train_preds)
```

Out[100]:

```
0.7866353916648483
```

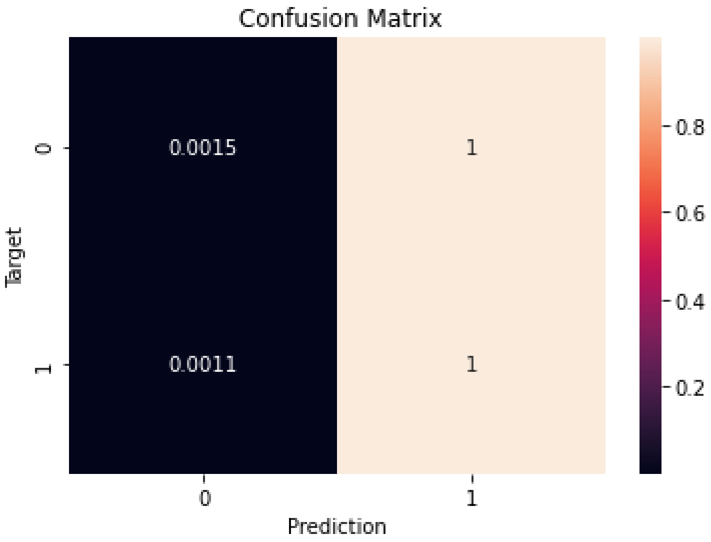
In [92]:

```
from sklearn.metrics import confusion_matrix

def predict_and_plot(inputs, targets, name=''):
    preds = model.predict(inputs)

    accuracy = accuracy_score(targets, preds)
    print("Accuracy: {:.2f}%".format(accuracy * 100))

    cf = confusion_matrix(targets, preds, normalize='true')
    plt.figure()
    sns.heatmap(cf, annot=True)
    plt.xlabel('Prediction')
    plt.ylabel('Target')
    plt.title('{} Confusion Matrix'.format(name));

    return preds, accuracy
```

In [93]:

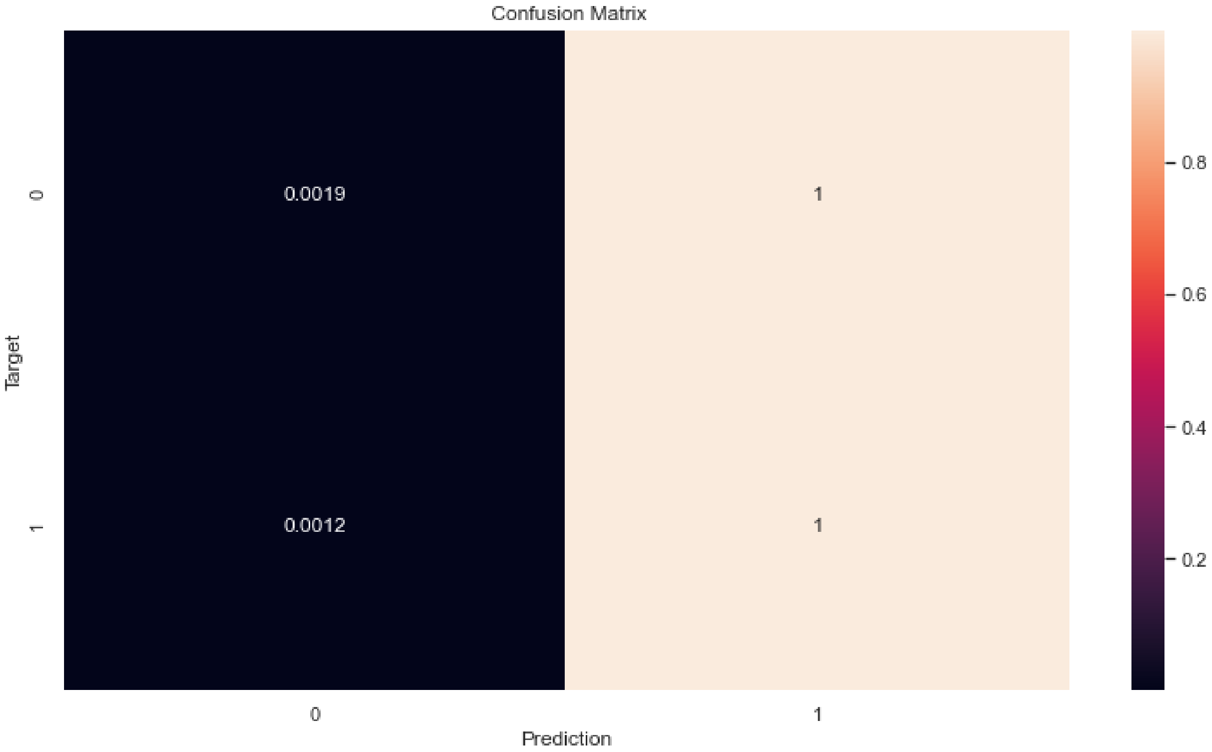
```
#Training Acuuracy
import seaborn as sns
train_preds = predict_and_plot(train_inputs, train_targets)
```

```
Accuracy: 78.66%
```



```
In [121]: #Testing Accuracy
          test_accuracy_lr = predict_and_plot(test_inputs, test_targets)
          scores_dict['Logistic Regression']=0.7802
```

```
Accuracy: 78.02%
```



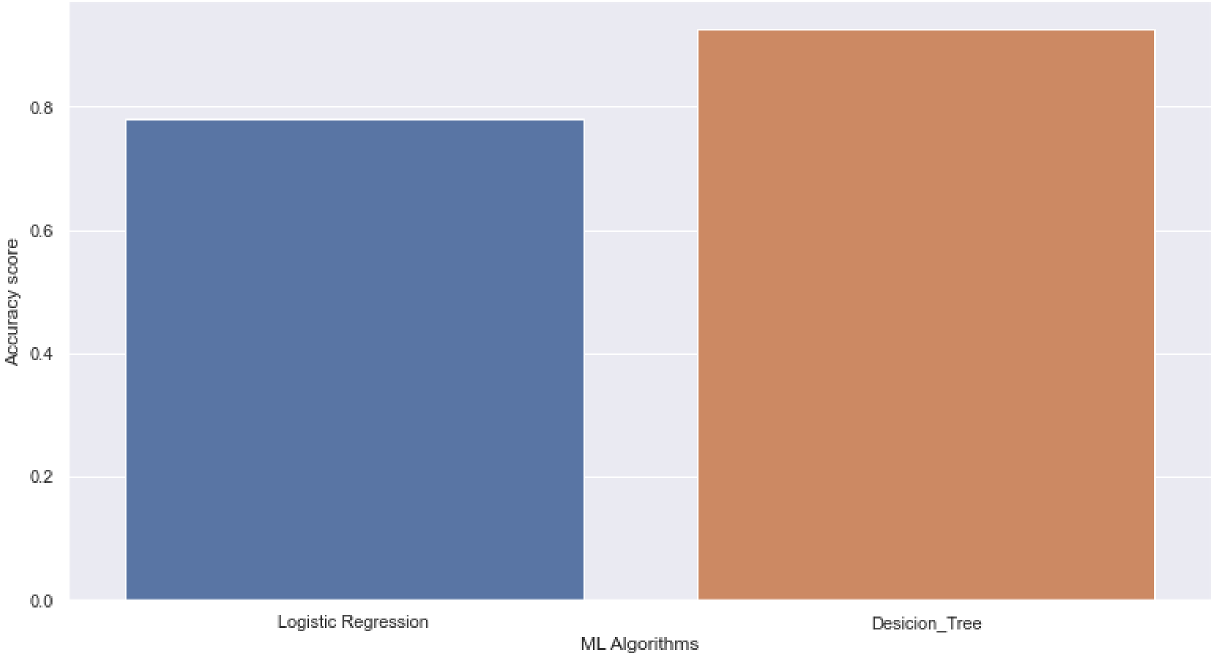
Implementing Desicion Tree

In [124]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100, m

clf_entropy.fit(train_inputs, train_targets)

y_pred = clf_entropy.predict(test_inputs)

test_accuracy_dt = metrics.precision_score(test_targets, y_pred)
print("Accuracy:",test_accuracy_dt)
print("Precision:",metrics.precision_score(test_targets, y_pred))
print("Recall:",metrics.recall_score(test_targets, y_pred))

clf_entropy.fit(val_inputs, val_targets)
y_pred = clf_entropy.predict(val_inputs)
validation_accuracy_dt = metrics.precision_score(val_targets, y_pred)
scores_dict['Desicion_Tree']=test_accuracy_dt
```

```
Accuracy: 0.926561709228686
Precision: 0.926561709228686
Recall: 0.9480136590099084
```

In [123]:
```
with sns.color_palette('muted'):
    algo_name = list(scores_dict.keys())
    scoress = list(scores_dict.values())

    sns.set(rc={'figure.figsize':(13,7)})
    plt.xlabel("ML Algorithms")
    plt.ylabel("Accuracy score")

    sns.barplot(algo_name,scoress)
```



Conclusion: Among the 2 Algorithms Desicion Tree has the highest accuracy.

In [ ]: