Kingdom of Saudi Arabia
The Ministry of Education
Prince Sattam bin Abdulaziz University
College of Computer Engineering and Sciences

المملكة العربية السعودية
وزارة التربية والتعليم
جامعة الأمير سطام بن عبد العزيز
كلية الهندسة وعلوم الحاسب

EDA, time series and climate regions analysis.

Student name :

Remas alqarni

Waad AlYahya

Reyouf Alhubaishi

Ghina Alamar

Project sponsor :

Fatma masmoudi

Contents:

What are the challenges of WIDS2023?

This year's Datathon focuses on utilizing data science to better prepare and adapt to the challenges caused by climate change and extreme weather events.

Participants are challenged to blend machine-learning and physics-based forecasts to improve long-term weather forecasting, thereby helping communities and industries adapt to the challenges of climate change.

Whether a beginner or experienced data scientist.

The WiDS Datathon dataset was created in collaboration with Climate Change AI (CCAI).

Goals :

The challenge was to participate in the competition.

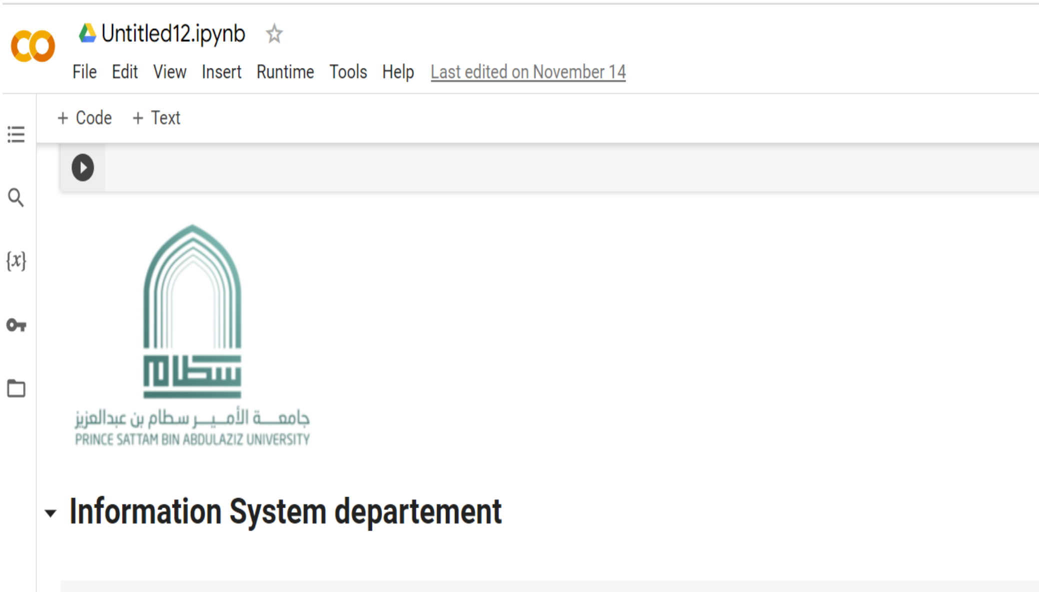
Our goal is to use various graphs and statistics over the years.

Using graphs and statistics we analyze climate zones and time series on targeted changes .

Project steps and Addition the code to Google Club

Step1:

In the beginning, the text icon was used to add two images



1- The university logo was added, and a short text for the department name was also added.

2- The image has been added that expresses different weather conditions and winds, and this explains the project title.

Step 2:

Loading the Data:

```
[ ] import numpy as np
    import pandas as pd
    from IPython.display import clear_output
    import os
    for dirname, _, filenames in os.walk('/content/submission (15).csv'):
        for filename in filenames:
            print(os.path.join(dirname, filename))

    !pip install missingno gdown
    !pip install
    clear_output(wait=False)

    import random
    import math
    import re
    from sklearn.impute import SimpleImputer
    import seaborn as sns
    import matplotlib.pyplot as plt
    import matplotlib.colors
    import plotly.express as px
    import missingno as msno
    pd.set_option('display.max_rows', 500)
    pd.set_option('display.max_columns', 500)

    VALIDATION_RATIO = 0.05
    RANDOM_SEED = 777
```

The data files were uploaded to the Google Colab program, from which we extract the information we need, Upload the data to the files icon and download.

A code was written to show us how to download the data and identify it so that extracting the data would be easier.

Step 3:

Reading the Data:

```
[ ] train_df = pd.read_csv('/content/submission (15).csv')
    test_df = pd.read_csv('/content/submission (15).csv')
    Target = 'contest-tmp2m-14d__tmp2m'
```

In the Google Colab program, from the code icon, three programming statements are written, and these statements explain to us as follows:

1-The df statement is used to define a new function, which is

1- train_df .

2- test_df .

2-Training/testing is a way to measure model accuracy.

It is called training/testing because the data set is split into two sets:

Example: training set and test set.

3-"target" is just a variable name. You can use any other variable name instead of "target" and it won't make any difference.

Step 4:

Read the Data:

Output:

```
[ ] train_df
```

	index	lat	lon	startdate	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	nmme0-tmp2m-34w__cancm40	nmme0-tmp2m-34w__ccsm30	nmme0-tmp2m-34w__ccsm40	nmme0-tmp2m-34w__cfsv20	nmme0-tmp2m-34w__gfdlflora0	nmme0-tmp2m-34w__gfdlflorb0	nmme0-tmp2m-34w__gfdl0	34
0	0	0.000000	0.833333	9/1/14	237.00	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56	
1	1	0.000000	0.833333	9/2/14	228.90	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56	
2	2	0.000000	0.833333	9/3/14	220.69	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56	
3	3	0.000000	0.833333	9/4/14	225.28	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56	
4	4	0.000000	0.833333	9/5/14	237.24	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4568	4568	0.090909	0.833333	3/2/15	175.39	9.63	12.49	12.58	13.05	10.77	13.41	14.16	10.58	
4569	4569	0.090909	0.833333	3/3/15	186.61	9.63	12.49	12.58	13.05	10.77	13.41	14.16	10.58	
4570	4570	0.090909	0.833333	3/4/15	189.45	9.63	12.49	12.58	13.05	10.77	13.41	14.16	10.58	
4571	4571	0.090909	0.833333	3/5/15	188.66	9.63	12.49	12.58	13.05	10.77	13.41	14.16	10.58	
4572	4572	0.090909	0.833333	3/6/15	180.84	9.63	12.49	12.58	13.05	10.77	13.41	14.16	10.58	

4573 rows × 246 columns

wind-vwnd-250-2010-9	wind-vwnd-250-2010-10	wind-vwnd-250-2010-11	wind-vwnd-250-2010-12	wind-vwnd-250-2010-13	wind-vwnd-250-2010-14	wind-vwnd-250-2010-15	wind-vwnd-250-2010-16	wind-vwnd-250-2010-17	wind-vwnd-250-2010-18	wind-vwnd-250-2010-19	wind-vwnd-250-2010-20	wind-uwnd-250-2010-1	wi uw 2 20
15.14	-99.89	7.88	5.91	-208.23	18.67	21.00	134.88	43.65	-44.70	-3.70	-65.02	628.66	130
-2.39	-113.06	1.33	17.87	-206.98	23.89	5.08	139.95	45.29	-37.26	3.63	-50.56	615.58	135
-20.02	-118.37	-8.50	24.55	-194.54	28.53	-15.99	140.68	42.80	-27.63	9.96	-32.91	602.14	142
-37.61	-114.71	-13.53	26.21	-173.41	31.15	-34.71	139.34	41.75	-18.27	20.78	-19.75	589.63	149
-53.54	-104.17	-19.09	24.10	-151.92	34.04	-38.14	137.02	40.43	-10.92	27.23	-7.78	576.23	159
...	...	...	...	...	...	...	...	...	...	...	...	...	
149.97	-87.62	-32.91	44.76	-66.13	-82.01	-41.94	-49.85	97.01	-67.00	52.33	32.64	-489.86	-77
159.58	-82.31	-35.48	51.66	-72.74	-76.17	-8.60	-45.62	102.57	-69.50	47.40	43.47	-468.78	-76
150.21	-71.44	-43.18	57.70	-82.34	-64.92	-7.45	-54.95	122.77	-70.40	48.70	56.55	-451.23	-84
144.86	-55.48	-41.95	56.55	-80.80	-54.92	-23.26	-57.15	156.25	-68.89	46.15	65.75	-437.45	-96
152.56	-44.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

Step 5:

Missing values and their correction:

The data is presented clearly and each column shows us the numbers and dates.

With the same function, the data table was displayed without errors, because we noticed in the previous table that some data had disappeared and NaN was written in its place.

Therefore, the errors were corrected using the dropna() function.

Output:

```
train_df.dropna()
```

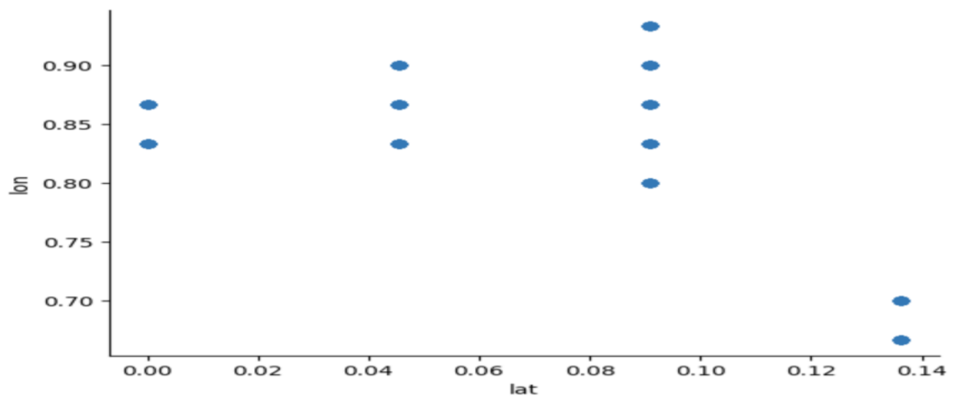
	index	lat	lon	startdate	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	nmme0-tmp2m-34w__cancm40	nmme0-tmp2m-34w__ccsm30	nmme0-tmp2m-34w__ccsm40	nmm tmp 34w__cfs
0	0	0.000000	0.833333	9/1/14	237.00	29.02	31.64	29.57	30.73	2
1	1	0.000000	0.833333	9/2/14	228.90	29.02	31.64	29.57	30.73	2
2	2	0.000000	0.833333	9/3/14	220.69	29.02	31.64	29.57	30.73	2
3	3	0.000000	0.833333	9/4/14	225.28	29.02	31.64	29.57	30.73	2
4	4	0.000000	0.833333	9/5/14	237.24	29.02	31.64	29.57	30.73	2
...	...	...	...	...	...	...	...	...	...	
45890	45890	0.272727	0.566667	3/22/16	575.41	1.95	4.76	3.29	2.72	
45891	45891	0.272727	0.566667	3/23/16	564.37	7.82	8.63	7.65	8.78	
45892	45892	0.272727	0.566667	3/24/16	539.64	7.82	8.63	7.65	8.78	
45893	45893	0.272727	0.566667	3/25/16	548.03	7.82	8.63	7.65	8.78	
45894	45894	0.272727	0.566667	3/26/16	548.00	7.82	8.63	7.65	8.78	

45895 rows × 246 columns

Step 6:

Add a prediction:

```
from matplotlib import pyplot as plt
_df_6.plot(kind='scatter', x='lat', y='lon', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```



We used point representation for prediction.

Step 7:

Read the Data:

```
test_df
```

	index	lat	lon	startdate	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	nmme0-tmp2m-34w__cancm40	nmme0-tmp2m-34w__ccsm30	nmme0 tmp2m 34w__ccsm4
0	0	0.000000	0.833333	9/1/14	237.00	29.02	31.64	29.57	30.7
1	1	0.000000	0.833333	9/2/14	228.90	29.02	31.64	29.57	30.7
2	2	0.000000	0.833333	9/3/14	220.69	29.02	31.64	29.57	30.7
3	3	0.000000	0.833333	9/4/14	225.28	29.02	31.64	29.57	30.7
4	4	0.000000	0.833333	9/5/14	237.24	29.02	31.64	29.57	30.7
...	...	...	...	...	...	...	...	...	.
46550	46550	0.272727	0.600000	1/11/16	191.88	0.07	1.87	2.23	1.6
46551	46551	0.272727	0.600000	1/12/16	209.69	0.07	1.87	2.23	1.6
46552	46552	0.272727	0.600000	1/13/16	218.53	0.07	1.87	2.23	1.6
46553	46553	0.272727	0.600000	1/14/16	221.86	0.07	1.87	2.23	1.6
46554	46554	0.272727	0.600000	1/15/16	222.01	0.07	1.87	2.23	1.6

46555 rows × 246 columns

Every time the spreadsheet is displayed, we make sure that there are no missing values, but we notice that there are missing values that we have to correct.

```
test_df
```

wind- t-10- 010-6	wind- hgt-10- 2010-7	wind- hgt-10- 2010-8	wind- hgt- 10- 2010-9	wind- hgt-10- 2010-10	wind- hgt-100- 2010-1	wind- hgt- 100- 2010-2	wind- hgt- 100- 2010-3	wind- hgt- 100- 2010-4	wind- hgt- 100- 2010-5	wind- hgt- 100- 2010-6	wind- hgt- 100- 2010-7	win hg 10 2010
81.45	-6076.15	-2209.63	3864.18	-3051.21	-25749.70	-5160.59	-1507.91	3391.32	-288.52	-1585.41	1544.02	944.
57.36	-6672.23	-3786.46	2626.55	-3623.29	-25474.37	-5356.70	-1367.76	3188.99	-221.06	-1193.63	1256.48	2018.
47.08	-7296.78	-5193.92	1591.47	-4080.94	-25200.29	-5546.88	-1230.46	2996.82	-111.60	-796.13	936.58	2959.
43.65	-7883.22	-6267.81	816.90	-4445.18	-24789.70	-5692.21	-1177.18	2799.89	-38.07	-362.72	608.32	3796.
88.12	-8267.89	-7134.70	227.75	-4620.95	-24181.96	-5754.12	-1208.87	2582.56	-35.19	80.43	355.94	4507.
...	...	...	...	...	...	...	...	...	...	...	...	
37.45	-485.43	-915.80	2382.35	1552.23	37941.40	1012.62	1100.08	5758.90	366.65	-1351.20	-2619.82	-1161.
99.33	67.86	-1369.43	2842.30	1470.71	37930.58	636.21	1316.88	5817.58	371.61	-1217.70	-1928.67	-982.
45.70	670.38	-1719.99	3181.43	1395.68	37940.23	108.66	1490.74	5957.24	369.78	-1144.29	-1274.44	-776.
61.17	1387.01	-2069.35	3429.98	1276.14	37913.47	-452.36	1639.97	6107.55	344.00	-1118.80	-676.08	-498.
83.17	2237.84	-2580.48	3738.07	1251.50	37849.63	-936.32	1789.85	6218.11	244.74	NaN	NaN	N

Step 8:

Missing values and their correction:

The data is presented clearly and each column shows us the numbers and dates.

With the same function, the data table was displayed without errors, because we noticed in the previous table that some data had disappeared and NaN was written in its place.

Therefore, the errors were corrected using the dropna() function.

Output:

```
test_df.dropna()
```

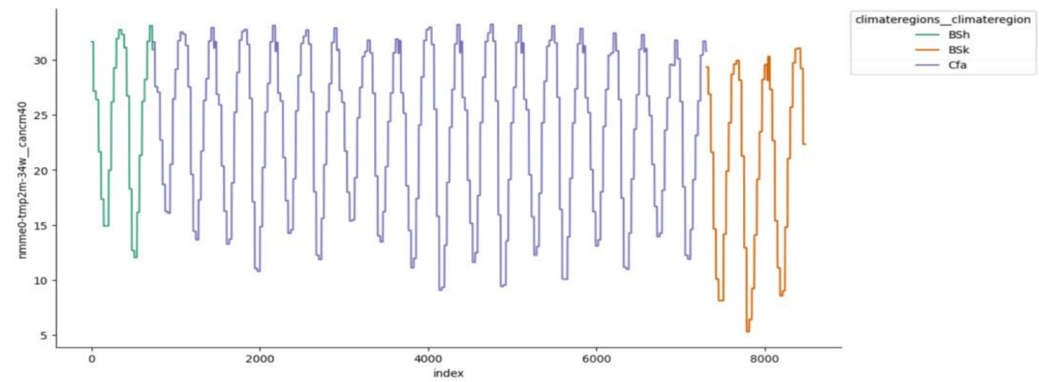
	index	lat	lon	startdate	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	nmme0-tmp2m-34w__cancm40	nmme0-tmp2m-34w__ccsm30	nmme0-tmp2m-34w__ccsm40	nmme0-tmp2m-34w__cfsv20	nmme0-tmp2m-34w__gfdlflora0	nmme0-tmp2m-34w__gfdlflorb0	nmme0-tmp2m-34w__gfdl0
0	0	0.000000	0.833333	9/1/14	237.00	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
1	1	0.000000	0.833333	9/2/14	228.90	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
2	2	0.000000	0.833333	9/3/14	220.69	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
3	3	0.000000	0.833333	9/4/14	225.28	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
4	4	0.000000	0.833333	9/5/14	237.24	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Step 9:

Add a prediction

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['index']
  ys = series['nmme0-tmp2m-34w__cancm40']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_33.sort_values('index', ascending=True)
for i, (series_name, series) in enumerate(df_sorted.groupby('climateregions__climateregion')):
  _plot_series(series, series_name, i)
  fig.legend(title='climateregions__climateregion', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('index')
_ = plt.ylabel('nmme0-tmp2m-34w__cancm40')
```


```
  _plot_series(series, series_name, i)
  fig.legend(title='climateregions__climateregion', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('index')
_ = plt.ylabel('nmme0-tmp2m-34w__cancm40')
```

Use both functions:

1-train_df.info().

2-test_df.info().

To display data in the form of information such as a paragraph, including the number of columns, range, and type.

Output:

```
[42] train_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16985 entries, 0 to 16984
Columns: 246 entries, index to wind-vwnd-925-2010-20
dtypes: float64(239), int64(4), object(3)
memory usage: 31.9+ MB
```

```
test_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16985 entries, 0 to 16984
Columns: 246 entries, index to wind-vwnd-925-2010-20
dtypes: float64(239), int64(4), object(3)
memory usage: 31.9+ MB
```

Step 10:

View other data in detail. There are no missing values.

```
train_df.describe()
```

	index	lat	lon	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	nmme0-tmp2m-34w__cancm40	nmme0-tmp2m-34w__ccsm30	nmme0-tmp2m-34w__ccsm40	nmme0-tmp2m-34w__cfsv20	nm 34w__
count	45896.000000	45896.000000	45896.000000	45896.000000	45896.000000	45896.00000	45896.000000	45896.000000	45896.000000	45
mean	22947.500000	0.183679	0.709482	329.802769	18.688813	20.67796	17.476635	19.130578	17.948764	
std	13249.178314	0.070734	0.206541	155.819879	9.341623	8.70244	8.505720	8.176927	7.597761	
min	0.000000	0.000000	0.233333	43.330000	-0.340000	1.54000	-0.550000	-0.660000	0.350000	
25%	11473.750000	0.136364	0.533333	210.970000	10.340000	12.98000	9.670000	12.180000	11.410000	
50%	22947.500000	0.181818	0.766667	296.285000	19.870000	21.69000	17.420000	20.180000	18.410000	
75%	34421.250000	0.227273	0.866667	438.127500	27.580000	28.60000	25.380000	26.620000	24.840000	
max	45895.000000	0.272727	1.000000	986.870000	36.080000	35.13000	33.260000	34.640000	35.750000	

Step 11:

Checking & Imputing Missing Values

```
def check_null_index(df):
    null_check_df = df.isnull().any()
    non_null_index_list = list((null_check_df[null_check_df==False]).index)
    null_index_list = list((null_check_df[null_check_df==True]).index)
    print(non_null_index_list)
    print(null_index_list)

    return null_index_list
```

Output:

```
train_null_col = check_null_index(train_df)
```

```
['index', 'lat', 'lon', 'startdate', 'contest-pevpr-sfc-gauss-
['wind-hgt-10-2010-6', 'wind-hgt-10-2010-7', 'wind-hgt-10-2010
```

```
[19] test_null_col = check_null_index(test_df)
```

```
['index', 'lat', 'lon', 'startdate', 'contest-pevpr-sfc-gauss-14d__
['wind-hgt-10-2010-6', 'wind-hgt-10-2010-7', 'wind-hgt-10-2010-8',
```

Step 12:

Show all columns.

Output:

```
train_null_col
```

```
['wind-hgt-10-2010-6',
 'wind-hgt-10-2010-7',
 'wind-hgt-10-2010-8',
 'wind-hgt-10-2010-9',
 'wind-hgt-10-2010-10',
 'wind-hgt-100-2010-1',
 'wind-hgt-100-2010-2',
 'wind-hgt-100-2010-3',
 'wind-hgt-100-2010-4',
 'wind-hgt-100-2010-5',
 'wind-hgt-100-2010-6',
 'wind-hgt-100-2010-7',
 'wind-hgt-100-2010-8',
 'wind-hgt-100-2010-9',
 'wind-hgt-100-2010-10',
 'wind-vwnd-925-2010-1',
 'wind-vwnd-925-2010-2',
```

Step 13:

Show all columns.

Output:

```
test_null_col
```

```
['wind-hgt-10-2010-6',
 'wind-hgt-10-2010-7',
 'wind-hgt-10-2010-8',
 'wind-hgt-10-2010-9',
 'wind-hgt-10-2010-10',
 'wind-hgt-100-2010-1',
 'wind-hgt-100-2010-2',
 'wind-hgt-100-2010-3',
 'wind-hgt-100-2010-4',
 'wind-hgt-100-2010-5',
 'wind-hgt-100-2010-6',
 'wind-hgt-100-2010-7',
 'wind-hgt-100-2010-8',
 'wind-hgt-100-2010-9',
 'wind-hgt-100-2010-10',
 'wind-vwnd-925-2010-1',
 'wind-vwnd-925-2010-2',
 'wind-vwnd-925-2010-3',
 'wind-vwnd-925-2010-4',
 'wind-vwnd-925-2010-5',
```

Step 14:

We notice that there are only columns containing missing values.

Output:

```
train_df[train_null_col]
```

	wind-hgt-10-2010-6	wind-hgt-10-2010-7	wind-hgt-10-2010-8	wind-hgt-10-2010-9	wind-hgt-10-2010-10	wind-hgt-100-2010-1	wind-hgt-100-2010-2	w 2(
0	-2281.45	-6076.15	-2209.63	3864.18	-3051.21	-25749.70	-5160.59	-15
1	-1957.36	-6672.23	-3786.46	2626.55	-3623.29	-25474.37	-5356.70	-13
2	-1747.08	-7296.78	-5193.92	1591.47	-4080.94	-25200.29	-5546.88	-12
3	-1643.65	-7883.22	-6267.81	816.90	-4445.18	-24789.70	-5692.21	-11
4	-1488.12	-8267.89	-7134.70	227.75	-4620.95	-24181.96	-5754.12	-12
...	...	...	...	...	...	...	...	
16980	-1011.68	810.02	-866.62	3589.16	324.04	37075.00	-3467.59	-22
16981	-1360.14	1131.52	-1002.04	3613.74	338.88	36941.15	-3697.58	-24
16982	-1827.81	1563.86	-1314.23	3749.90	408.81	36870.88	-4021.84	-27
16983	-2216.56	2070.89	-1806.19	3930.81	471.68	36779.27	-4210.63	-28
16984	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Step 15:

We will address the problem of missing values by deletion:

Output:

```
train_df[train_null_col].dropna()
```

	wind-hgt-10-2010-6	wind-hgt-10-2010-7	wind-hgt-10-2010-8	wind-hgt-10-2010-9	wind-hgt-10-2010-10	wind-hgt-100-2010-1	wind-hgt-100-2010-2	wind-hgt-100-2010-3
0	-2281.45	-6076.15	-2209.63	3864.18	-3051.21	-25749.70	-5160.59	-1507.91
1	-1957.36	-6672.23	-3786.46	2626.55	-3623.29	-25474.37	-5356.70	-1367.76
2	-1747.08	-7296.78	-5193.92	1591.47	-4080.94	-25200.29	-5546.88	-1230.46
3	-1643.65	-7883.22	-6267.81	816.90	-4445.18	-24789.70	-5692.21	-1177.18
4	-1488.12	-8267.89	-7134.70	227.75	-4620.95	-24181.96	-5754.12	-1208.87
...	...	...	...	...	...	...	...	...
16979	-889.23	586.22	-863.32	3685.07	341.14	37357.20	-3397.19	-2009.82
16980	-1011.68	810.02	-866.62	3589.16	324.04	37075.00	-3467.59	-2233.69
16981	-1360.14	1131.52	-1002.04	3613.74	338.88	36941.15	-3697.58	-2496.94
16982	-1827.81	1563.86	-1314.23	3749.90	408.81	36870.88	-4021.84	-2729.99
16983	-2216.56	2070.89	-1806.19	3930.81	471.68	36779.27	-4210.63	-2898.69

16984 rows × 35 columns

Step 16:

```python
def impute_number_col(df):
    null_col = ['nmme0-tmp2m-34w__ccsm30',
    'nmme-tmp2m-56w__ccsm3',
    'nmme-prate-34w__ccsm3',
    'nmme0-prate-56w__ccsm30',
    'nmme0-prate-34w__ccsm30',
    'nmme-prate-56w__ccsm3',
    'nmme-tmp2m-34w__ccsm3',
    'ccsm30']
    number_imputer = SimpleImputer(missing_values=np.nan, strategy='median')
    fixed_column_df = number_imputer.fit_transform(df[null_col])
    df[null_col] = fixed_column_df
    return df
```

Step 17:

```python
[25] nonnull_train_df = impute_number_col(train_df)
```

Step 18:

View an data table.

Output:

```python
nonnull_train_df
```

	index	lat	lon	startdate	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	nmm 34w
0	0	0.000000	0.833333	9/1/14	237.00	29.02	
1	1	0.000000	0.833333	9/2/14	228.90	29.02	
2	2	0.000000	0.833333	9/3/14	220.69	29.02	
3	3	0.000000	0.833333	9/4/14	225.28	29.02	
4	4	0.000000	0.833333	9/5/14	237.24	29.02	
...	...	...	...	...	...	...	
16980	16980	0.181818	0.700000	2/15/15	278.48	2.90	
16981	16981	0.181818	0.700000	2/16/15	276.15	2.90	
16982	16982	0.181818	0.700000	2/17/15	273.23	2.90	
16983	16983	0.181818	0.700000	2/18/15	271.96	2.90	
16984	16984	0.181818	0.700000	2/19/15	269.48	2.90	

```python
nonnull_train_df
```

wind-hgt-100-2010-2	wind-hgt-100-2010-3	wind-hgt-100-2010-4	wind-hgt-100-2010-5	wind-hgt-100-2010-6	wind-hgt-100-2010-7	wind-hgt-100-2010-8
5160.59	-1507.91	3391.32	-288.52	-1585.41	1544.02	944.73
5356.70	-1367.76	3188.99	-221.06	-1193.63	1256.48	2018.62
5546.88	-1230.46	2996.82	-111.60	-796.13	936.58	2959.85
5692.21	-1177.18	2799.89	-38.07	-362.72	608.32	3796.72
5754.12	-1208.87	2582.56	-35.19	80.43	355.94	4507.20
...	...	...	...	...	...	...
3467.59	-2233.69	7633.09	716.76	-2140.14	-2114.48	214.93
3697.58	-2496.94	7588.55	933.87	-1731.48	-2011.97	171.06
4021.84	-2729.99	7604.67	1100.60	-1393.45	-1914.31	109.87
4210.63	-2898.69	7608.69	1298.51	-1167.18	-1715.76	140.85
NaN	NaN	NaN	NaN	NaN	NaN	NaN

Step 19:

Correct missing values.

Output:

```
nonnull_train_df.dropna()
```

	index	lat	lon	startdate	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	n 3
0	0	0.000000	0.833333	9/1/14	237.00	29.02	
1	1	0.000000	0.833333	9/2/14	228.90	29.02	
2	2	0.000000	0.833333	9/3/14	220.69	29.02	
3	3	0.000000	0.833333	9/4/14	225.28	29.02	
4	4	0.000000	0.833333	9/5/14	237.24	29.02	
...	...	...	...	...	...	...	
16979	16979	0.181818	0.700000	2/14/15	278.37	2.90	
16980	16980	0.181818	0.700000	2/15/15	278.48	2.90	
16981	16981	0.181818	0.700000	2/16/15	276.15	2.90	
16982	16982	0.181818	0.700000	2/17/15	273.23	2.90	
16983	16983	0.181818	0.700000	2/18/15	271.96	2.90	

16984 rows × 246 columns

Step 20:

Create a table containing a group of different regions.

	climateregions__climateregion
BSk	139621
Dfb	52632
Cfa	51901
Csb	40936
Dfa	22661
BWk	13889
Dfc	12427
BWh	9503
Csa	9503
Dsb	8041
BSh	5117
Cfb	4386
Dsc	2924
Dwa	1462
Dwb	731

Output:

```
nonnull_train_df['climateregions__climateregion'].value_counts().to_frame()
```

	climateregions__climateregion
Cfa	1229
BSh	731

In which:

BSh: Hot semi-arid climate

BSk: Cold semi-arid climate

BWh: Hot desert climate

BWk: Cold desert climate

Cfa: Humid subtropical climate

Cfb: Temperate oceanic climate or subtropical highland climate

Csa: Hot-summer Mediterranean climate

Csb: Warm-summer Mediterranean climate

Dfa: Hot-summer humid continental climate

Dfb: Warm-summer humid continental climate

Dfc: Subarctic climate

Dsb: Mediterranean-influenced warm-summer humid continental climate

Dsc: Mediterranean-influenced subarctic climate

Dwa: Monsoon-influenced hot-summer humid continental climate

Dwb: Monsoon-influenced warm-summer humid continental climate

Prediction

Step 21:

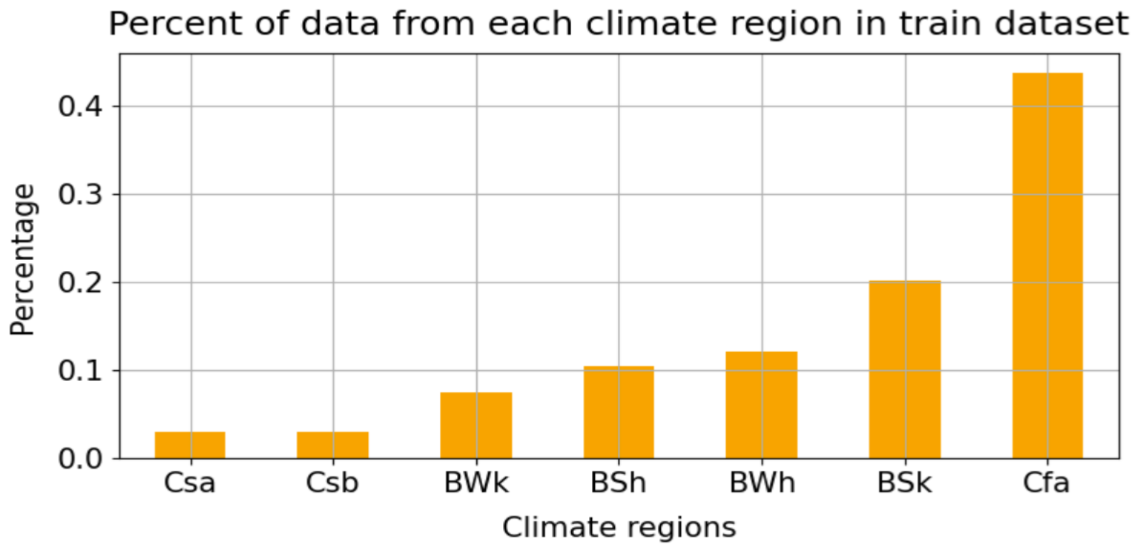
prediction

There is previous prediction of climate conditions in some areas

```
[ ] nonnull_train_df['climateregions__climateregion'].value_counts(normalize = True).sort_values().plot(kind='bar', color='Orange', figsize=(9,4), rot=0)

    plt.xlabel("Climate regions", labelpad=10, fontsize=15)
    plt.ylabel("Percentage", labelpad=10, fontsize=15)
    plt.xticks(size = 15)
    plt.yticks(size = 15)
    plt.grid()
    plt.title("Percent of data from each climate region in train dataset", y=1.02, fontsize=18)
```

Output:

```
Text(0.5, 1.02, 'Percent of data from each climate region in train dataset')
```



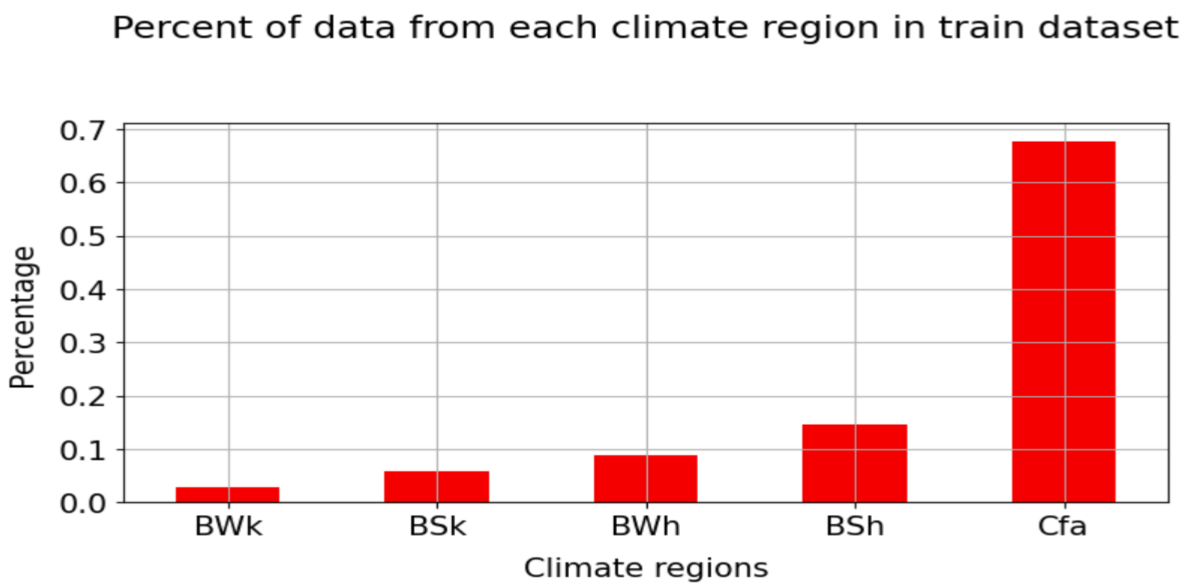
Step 22:

Prediction

Adding red color to the prediction chart

```
nonnull_train_df['climateregions__climateregion'].value_counts(normalize = True).sort_values().plot(kind='bar', color='red', figsize=(9,4), rot=0)

plt.xlabel("Climate regions", labelpad=10, fontsize=15)
plt.ylabel("Percentage", labelpad=10, fontsize=15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.grid()
plt.title("Percent of data from each climate region in train dataset", y=1.02, fontsize=18)
```

Output:

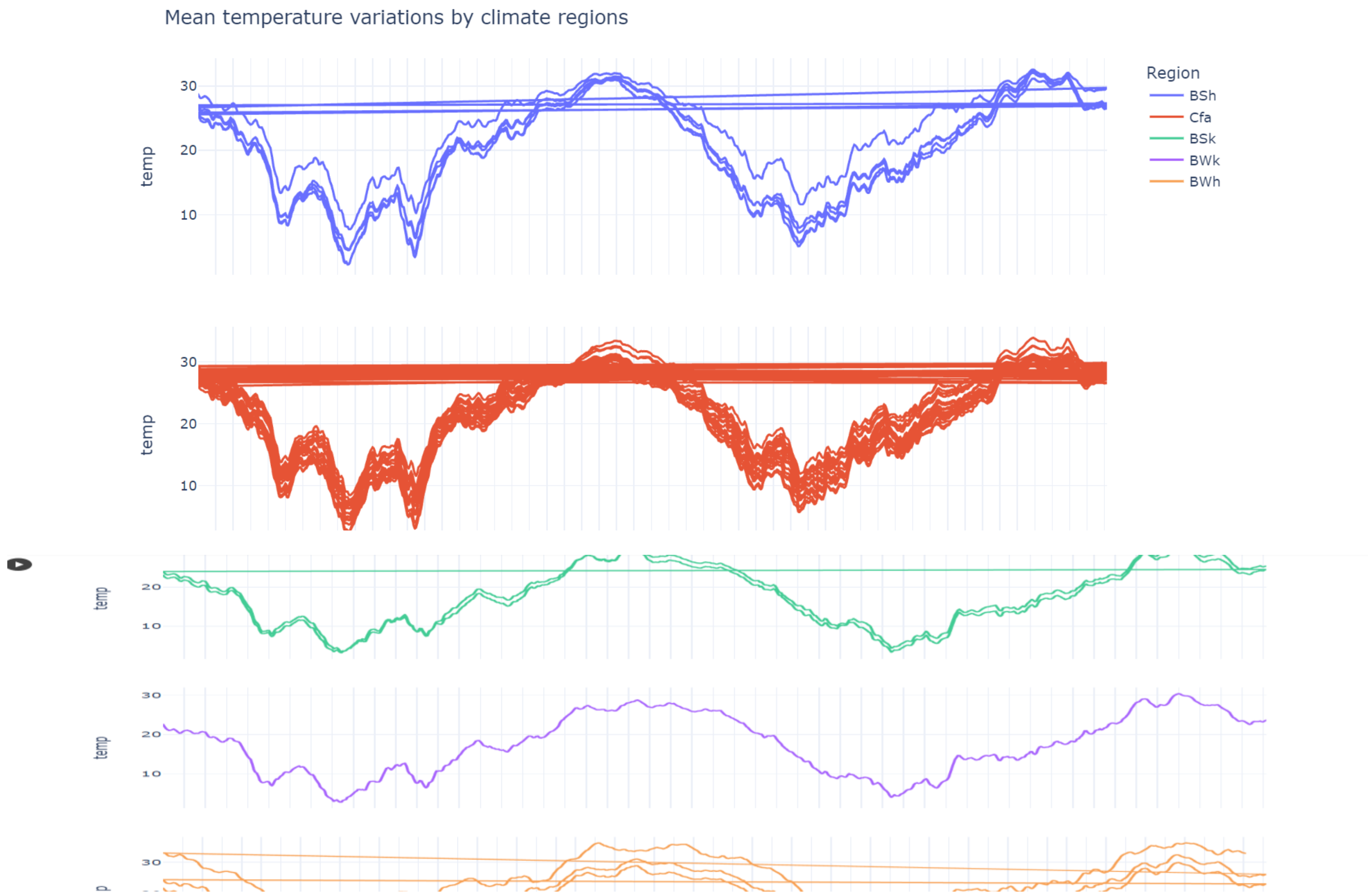


Step 23:

Prediction

```python
fig = px.line(nonnull_train_df, x='startdate',
              y='contest-tmp2m-14d__tmp2m',
              color = 'climateregions__climateregion',
              facet_row='climateregions__climateregion',facet_row_spacing=0.04,
              labels={"contest-tmp2m-14d__tmp2m":"temp", "climateregions__climateregion":"Region"},
              template = 'plotly_white', width=1000, height=1300)

fig.update_layout(title='Mean temperature variations by climate regions',
                  xaxis_title='Date', )

fig.update_yaxes(visible=True, matches=None)
fig.update_layout(annotations=[], overwrite=True)

fig.show()
```

Output:



Step 24:

Remove missing values and give a column lon and lat of zero values.

```
train_df_filtered = nonnull_train_df[(nonnull_train_df['lat'] == nonnull_train_df['lat'][0]) & (nonnull_train_df['lon'] == nonnull_train_df['lon'][0])]
train_df_filtered
```

Output:

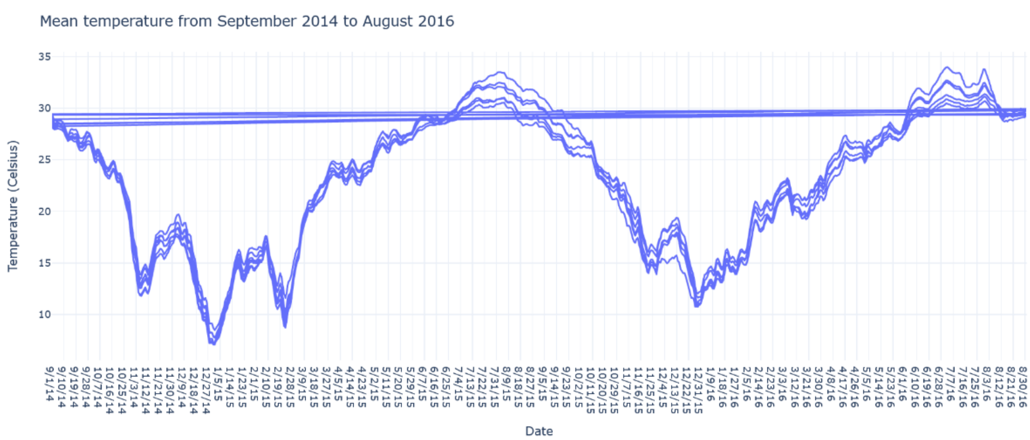
	index	lat	lon	startdate	contest-pevpr-sfc-gauss-14d__pevpr	nmme0-tmp2m-34w__cancm30	nmme0-tmp2m-34w__cancm40	nmme0-tmp2m-34w__ccsm30	nmme0-tmp2m-34w__ccsm40	nmme0-tmp2m-34w__cfsv20	nmme0-tmp2m-34w__gfdlflora0	nmme0-tmp2m-34w__gfdlflorb0	nmme0-tmp2m-34w__gfdl0
0	0	0.0	0.833333	9/1/14	237.00	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
1	1	0.0	0.833333	9/2/14	228.90	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
2	2	0.0	0.833333	9/3/14	220.69	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
3	3	0.0	0.833333	9/4/14	225.28	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
4	4	0.0	0.833333	9/5/14	237.24	29.02	31.64	29.57	30.73	29.71	31.52	31.68	30.56
...	...	...	...	...	...	...	...	...	...	...	...	...	...
726	726	0.0	0.833333	8/27/16	320.50	30.88	30.92	29.17	31.02	29.47	30.93	30.54	31.01
727	727	0.0	0.833333	8/28/16	325.39	30.88	30.92	29.17	31.02	29.47	30.93	30.54	31.01
728	728	0.0	0.833333	8/29/16	318.64	30.88	30.92	29.17	31.02	29.47	30.93	30.54	31.01
729	729	0.0	0.833333	8/30/16	319.93	30.88	30.92	29.17	31.02	29.47	30.93	30.54	31.01
730	730	0.0	0.833333	8/31/16	328.81	30.88	30.92	29.17	31.02	29.47	30.93	30.54	31.01

Step 25:

Clarity of temperatures in different years.

```
fig = px.line(nonnull_train_df, x='startdate', y=Target, template = 'plotly_white')

fig.update_layout(title='Mean temperature from September 2014 to August 2016',
                  xaxis_title='Date',
                  yaxis_title='Temperature (Celsius)')
fig.show()
```

Output:



Step 26:

Using 2D data, it contains columns and rows in a tabular manner.

```
[36] submission=pd.DataFrame(nonnull_train_df)
     submission.to_csv('submission.csv',index=False)
```

Output:

```
print(submission)
```

```
       index       lat       lon startdate  \
0          0  0.000000  0.833333    9/1/14
1          1  0.000000  0.833333    9/2/14
2          2  0.000000  0.833333    9/3/14
3          3  0.000000  0.833333    9/4/14
4          4  0.000000  0.833333    9/5/14
...      ...       ...       ...       ...
24835  24835  0.227273  0.366667   8/13/16
24836  24836  0.227273  0.366667   8/14/16
24837  24837  0.227273  0.366667   8/15/16
24838  24838  0.227273  0.366667   8/16/16
24839  24839  0.227273  0.366667   8/17/16

       contest-pevpr-sfc-gauss-14d__pevpr  nmme0-tmp2m-34w__cancm30  \
0                                  237.00                     29.02
1                                  228.90                     29.02
2                                  220.69                     29.02
3                                  225.28                     29.02
4                                  237.24                     29.02
...                                   ...                       ...
```

Step 27:

We added a simple code for 2D that does not have missing values

```
[ ] print(submission.dropna())
```

Output:

```
print(submission.dropna())
```

```
      index       lat       lon startdate  contest-pevpr-sfc-gauss-14d__pevpr  \
0         0  0.000000  0.833333    9/1/14                              237.00
1         1  0.000000  0.833333    9/2/14                              228.90
2         2  0.000000  0.833333    9/3/14                              220.69
3         3  0.000000  0.833333    9/4/14                              225.28
4         4  0.000000  0.833333    9/5/14                              237.24
...     ...       ...       ...       ...                                 ...
4567   4567  0.090909  0.833333    3/1/15                              164.93
4568   4568  0.090909  0.833333    3/2/15                              175.39
4569   4569  0.090909  0.833333    3/3/15                              186.61
4570   4570  0.090909  0.833333    3/4/15                              189.45
4571   4571  0.090909  0.833333    3/5/15                              188.66

      nmme0-tmp2m-34w__cancm30  nmme0-tmp2m-34w__cancm40  \
0                        29.02                     31.64
1                        29.02                     31.64
2                        29.02                     31.64
3                        29.02                     31.64
4                        29.02                     31.64
...                        ...                       ...
4567                      9.63                     12.49
4568                      9.63                     12.49
4569                      9.63                     12.49
4570                      9.63                     12.49
4571                      9.63                     12.49
```

Visualization

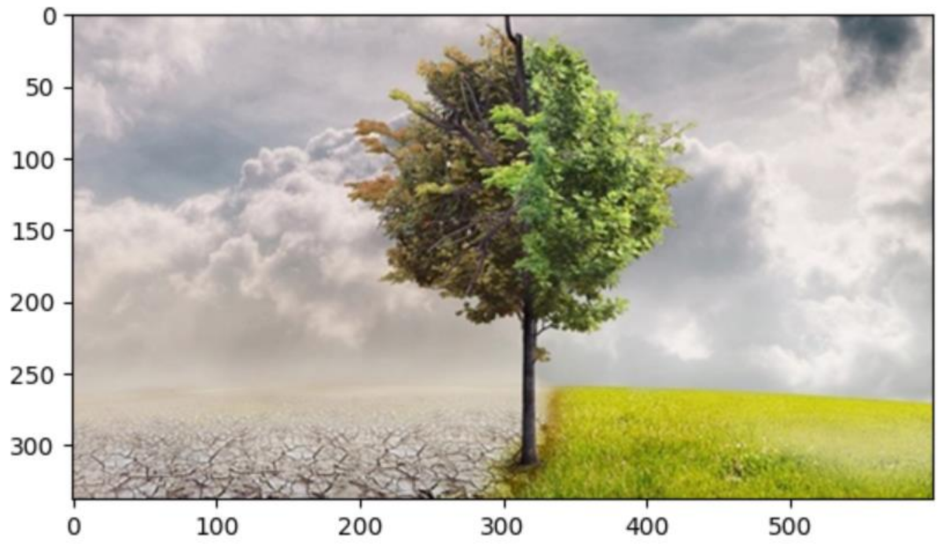
Step 28:

Visualization add

```
[ ] from scipy import misc
    import matplotlib.pyplot as plt
    import imageio

    image_path='/content/166.jpg'
    print(image_path)
    tree_image = imageio.imread(image_path)

    plt.imshow(tree_image)
    plt.show()
```

Output:

```
Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread.
```



Step 29:

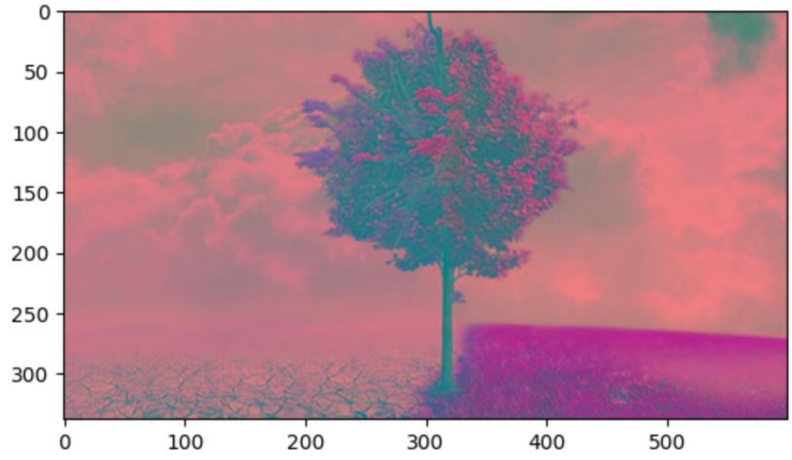
Visualization add

Visualization add and change the color of the image:

```
from scipy import misc
import matplotlib.pyplot as plt


tree_image_gray = imageio.imread(image_path, pilmode='YCbCr')

plt.imshow(tree_image_gray)
plt.show()
```

Output:

```
<ipython-input-38-a10cf4987001>:5: DeprecationWarning:

Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread.
```



Conclusion

In Conclusion, The WiDS 2023 competition serves as a tool for better comprehending the dataset.

In this notebook, we used statistical graphics and certain data visualization techniques and Prediction, we tried to evaluate and explore the training dataset in order to get its key characteristics and get a deeper understanding of its patterns and we fixed the errors that we found. and we analyze the impacts from the aspect of time series and climate regions on the variables.

Recommendations

We recommend participating in the challenge because it develops your skills in combining machine learning with physics-based visualizations and forecasts to improve long-term weather forecasting, and this is useful in helping communities and industries adapt to the challenges of climate change.