Data Preprocessing with Python

Machine Learning course, Chapter 3: Data Science, 5th Session
Project 1

SAMIRA SHEMIRANI #148

08/23/2023

Table of contents

01

Dataset

England Weather

What's in the Dataset?

The dataset on which we intend to perform pre-processing is called the "England Weather" dataset.

- 96453 data points have their information recorded in this dataset.
- 7 features have been examined for each data point.
- So, it can be said that this dataset is a table with 96453 rows and 7 columns.
- This dataset is 96453 × 7.

The format of this file is ".csv", and here, in the next column, we name the features of each column:

1. Formatted Date
2. Summary
3. Precip Type
4. Temperature (°C)
5. Wind Speed (km/h)
6. Pressure (millibars)
7. Humidity

02 Methods

Features and Histograms.

How to code for preprocessing?

Features Description (x axis)

Formatted Date

This column shows the *date*, *time*, and *coordinated universal time* zone (UTC) in the following format;
1st data point recorded on:
2006-04-01 00:00:00.000 +0200
Last data point recorded on:
2016-09-09 23:00:00.000 +0200

No Histogram Plot obtained.

Summary

This column shows the different weather conditions that have occurred. The number of 27 weather events that occurred, in descending order, is shown in the next page.

No Histogram Plot obtained.

Precip Type

The most common types of precipitation are rain and snow.

Rain	85,224
Snow	10,712
Null	517

No Histogram Plot obtained.

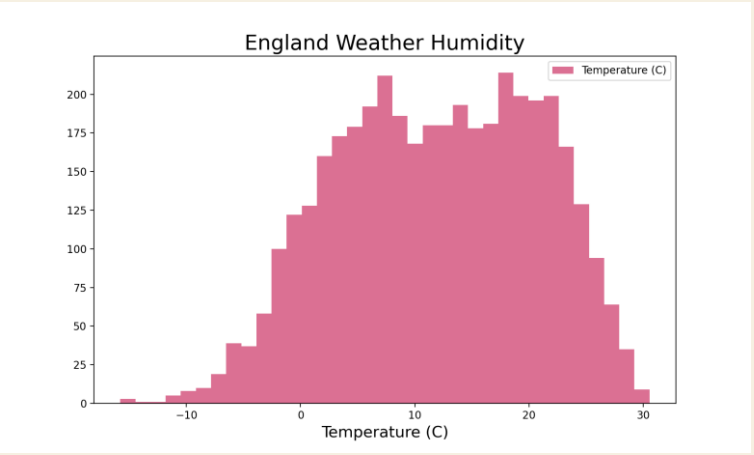
The Summary Table

Partly Cloudy	31,733
Mostly Cloudy	28,094
Overcast	16,597
Clear	10,890
Foggy	7,148
Breezy and Overcast	528
Breezy and Mostly Cloudy	516
Breezy and Partly Cloudy	386
Dry and Partly Cloudy	86
Windy and Partly Cloudy	67
Light Rain	63
Breezy	54
Windy and Overcast	45
Humid and Mostly Cloudy	40
Drizzle	39
Breezy and Foggy	35
Windy and Mostly Cloudy	35
Dry	34
Humid and Partly Cloudy	17
Dry and Mostly Cloudy	14
Rain	10
Windy	8
Humid and Overcast	7
Windy and Foggy	4
Breezy and Dry	1
Dangerously Windy and Partly Cloudy	1
Windy and Dry	1

Features Description (x axis)

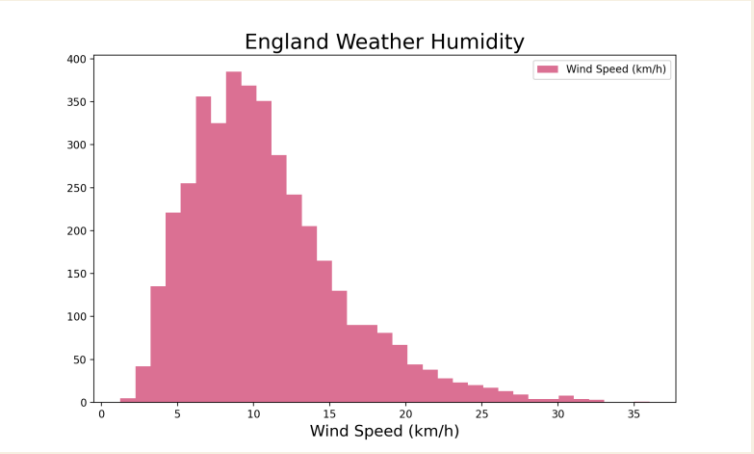
Temperature (°C)

The temperature in this column changes from -21.82 to 0 °C.
The height of the bar at the temperature of 7 °C and at the temperature of 18 °C, which are the highest heights of this histogram, shows us that in these two temperatures we have the highest value, so that maybe these two happened more than 200 times in this data set. The histogram of this feature shows us an asymmetric Gaussian, and in the middle of this Gaussian, we see a significant decrease in values.



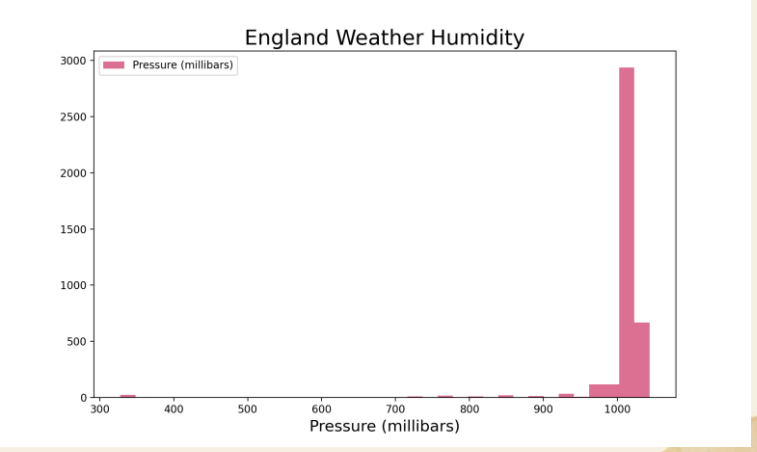
Wind Speed (km/h)

The wind speed in this column changes from 0 to 44.88 km/h.
The height of the bar at the wind speed 9 km/h, which is the highest height of this histogram, shows us that in this wind speed we have the highest value, so that maybe this happened more than 362 times in this data set. The histogram of this feature shows us an asymmetric Gaussian.



Pressure (millibars)

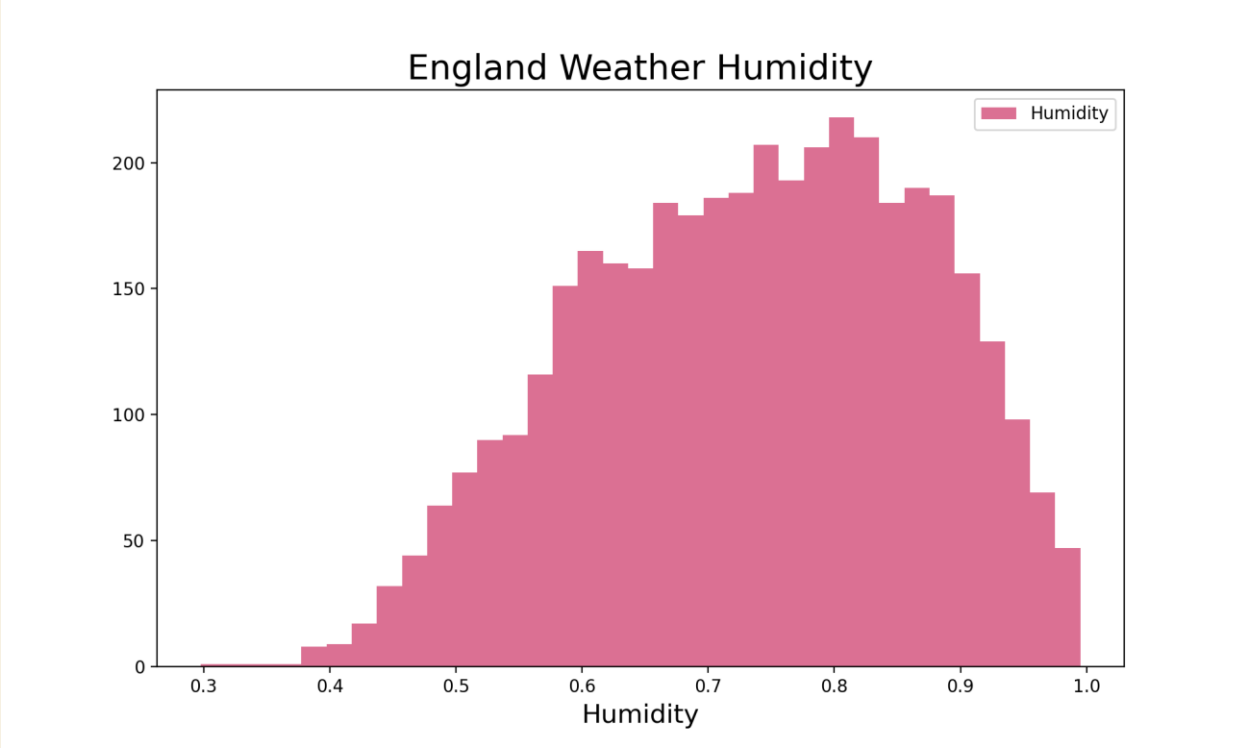
The pressure in this column changes from 0 to 1046.33 millibars.
The height of the bar at the pressure 1100 millibars, which is the highest height of this histogram, shows us that in this pressure we have the highest value, so that maybe this happened close to 3000 times in this data set.



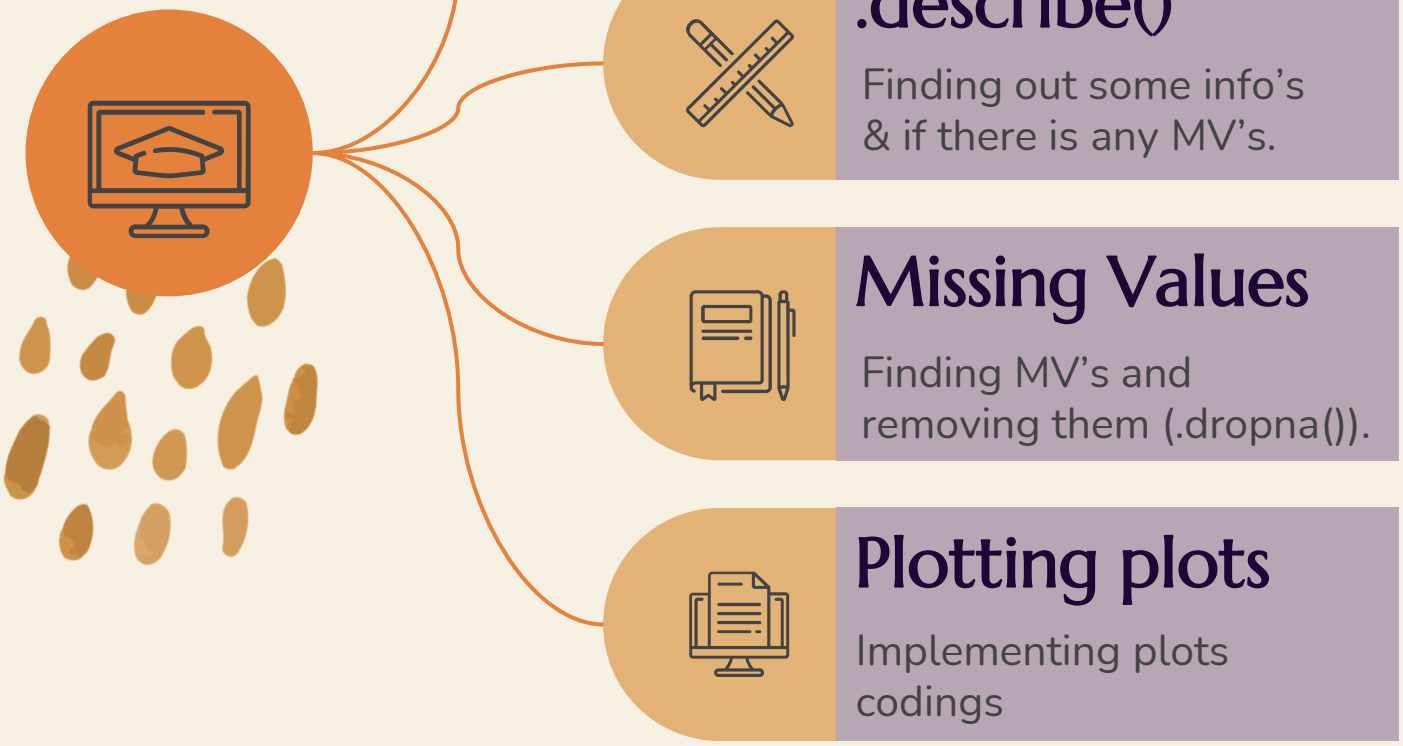
Target Description (y axis)

Humidity

The humidity in this column changes from 0 to 1.
The height of the bar at the humidity of 0.81, which are the highest height of this histogram, shows us that in this humidity we have the highest value, so that maybe this happened more than 200 times in this data set. The histogram of this feature shows us an asymmetric Gaussian.



What we've done in the coding part?



For the coding part, first we checked the dataset in Excel and Notepad++, and then we went to the Jupyter coding environment and did the following:

What's going on in our Jupyter Notebook?

Libraries

1 – We imported important libraries.

```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Dataset Reading

2 – We called the dataset into the coding environment, Jupyter Notebook & we made the DataFrame at the same time.

```
# reading the data into the coding enviroment
Data = pd.read_csv("D:/IMT/3- Data Science/5- Project #1/EnglandWeather.csv")
Data = pd.DataFrame(Data) # DataFrame with all the columns
Data
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Wind Speed (km/h)	Pressure (millibars)	Humidity
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	14.1197	1015.13	0.89
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	14.2646	1015.63	0.86
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	3.9284	1015.94	0.89
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	14.1036	1016.41	0.83
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	11.0446	1016.51	0.83
...	...	...	...	...	...	...	...

What's going on in our Jupyter Notebook?

Dataset Reading (cont.)

We have to do "*the splitting*" on the "Formatted Date". So here we go;

```
# trying to work with the 1st column which is time serie and spliting it.
# we don't want the 2nd (00:00:00.000) and 3rd (+0200) part of the 1st column,
# here is the process:

# two spaces are considered

new = Data["Formatted Date"].str.split(" ", n = 2, expand=True)
new
```

Out[4]:

	0	1	2
0	2006-04-01	00:00:00.000	+0200
1	2006-04-01	01:00:00.000	+0200
2	2006-04-01	02:00:00.000	+0200
3	2006-04-01	03:00:00.000	+0200
4	2006-04-01	04:00:00.000	+0200
...	...	...	...

Then we have to remove the basic column;

In [5]:

```
# removing the 1st column from the read dataset

Data1 = Data.drop(columns=["Formatted Date"])
Data1
```

Out[5]:

	Summary	Precip Type	Temperature (C)	Wind Speed (km/h)	Pressure (millibars)	Humidity
0	Partly Cloudy	rain	9.472222	14.1197	1015.13	0.89
1	Partly Cloudy	rain	9.355556	14.2646	1015.63	0.86
2	Mostly Cloudy	rain	9.377778	3.9284	1015.94	0.89
3	Partly Cloudy	rain	8.288889	14.1036	1016.41	0.83
4	Mostly Cloudy	rain	8.755556	11.0446	1016.51	0.83
...	...	...	...	...	...	...

What's going on in our Jupyter Notebook?

Dataset Reading (cont.)

Now is the time to insert the pre-made column into the DataFrame;

```
# now inserting the part that we want to be replaced instead of the "Formatted Date" column that we used to have

Data1.insert(0, "Date", new[0], True) # inplace=True
Data1

# here you can see that columns look fine!
```

	Date	Summary	Precip Type	Temperature (C)	Wind Speed (km/h)	Pressure (millibars)	Humidity
0	2006-04-01	Partly Cloudy	rain	9.472222	14.1197	1015.13	0.89
1	2006-04-01	Partly Cloudy	rain	9.355556	14.2646	1015.63	0.86
2	2006-04-01	Mostly Cloudy	rain	9.377778	3.9284	1015.94	0.89
3	2006-04-01	Partly Cloudy	rain	8.288889	14.1036	1016.41	0.83
4	2006-04-01	Mostly Cloudy	rain	8.755556	11.0446	1016.51	0.83
...	...	...	...	...	...	...	...

Now it's time to reduce (implementing mean) the number of recorded records of each day, which is about 23 or 24, to one record, because the recorded values of each day are close to each other (we are going to have 4018 rows);

```
# we have 24 ~ 23 rows per day, let's do something about it.
# first of all, sort the rows by the values of the "Date" column

# sorting the values inside the "Date" column
DataFrame = Data1.sort_values(by="Date")

# mean implementing of the 23~24 values of each day into only 1 value
DataMean = DataFrame.groupby(pd.Grouper(key="Date"), as_index=False).mean()
DataMean

# as you can see there are only the numeric columns left.
# the alphabetical columns were removed {"Summary" & "Precip Type"}
```

What's going on in our Jupyter Notebook?

Features & Target Recognition

3 – We decided which columns of Features are supposed to be our Target and which ones will remain as they are to be Features.

	Features				Target
	Date	Temperature (C)	Wind Speed (km/h)	Pressure (millibars)	Humidity
0	2006-01-01	3.873148	21.372750	1012.279167	0.818333
1	2006-01-02	5.418519	17.551683	1010.131667	0.844583
...	...	...	...	...	...
4016	2016-12-30	0.119444	10.806454	1020.395000	0.889167
4017	2016-12-31	0.072454	10.764862	1020.423750	0.888750

4018 rows × 5 columns

Creating DataFrame

4 – We made a DataFrame earlier in the cell that we read the dataset;

```
# reading the data into the coding enviroment
Data = pd.read_csv("D:/IMT/3- Data Science/5- Project #1/EnglandWeather.csv")
Data = pd.DataFrame(Data) # DataFrame with all the columns
Data
```

What's going on in our Jupyter Notebook?

.describe()

5 – We used to describe command to understand the statistical information of the dataset. Here we will find out if there is a Missing Value (MV) in the desired DataFrame (DataMean);

```
1 DataMean.describe() # describe in the numeric way of the numeric columns
```

	Temperature (C)	Wind Speed (km/h)	Pressure (millibars)	Humidity
count	4018.000000	4018.000000	4018.000000	4018.000000
mean	11.930135	10.812832	1003.233274	0.734882
std	8.778866	5.003314	71.325790	0.134333
min	-15.773611	1.245067	327.756800	0.297917
25%	5.046123	7.176575	1010.859167	0.632500
50%	12.245833	9.950806	1015.985208	0.743333
75%	19.269850	13.345894	1020.551979	0.842500
max	30.531481	36.002954	1043.574167	0.995000

There is no Missing Value
Correct!

Count shows; the number of rows/values in each column, if it does not have the same value as the rest, it means there is/are Missing Values. Here there is no MVs.
Mean shows: the mean of each column.
std shows: the standard deviation of each column.
Min & Max shows: the minimum and maximum values in each column.

What's going on in our Jupyter Notebook?

.describe()

And the rest of the non-numeric or alphabetic columns are described as follows;

```
# NOT IMPORTANT ***
Data.describe(include=object) # describe in the alphabetic way of the verbal columns
```

NOTE that we are .describe() 'ing Data, which has Missing Values and we don't have any Missing Value in our main DataFrame, which is "DataMean".

	Formatted Date	Summary	Precip Type
count	96453	96453	95936
unique	96429	27	2
top	2010-08-02 00:00:00.000 +0200	Partly Cloudy	rain
freq	2	31733	85224

Here you can see that, there is a Missing Value.

Count shows; the number of rows/values in each column, if it does not have the same value as the rest, it means there is/are Missing Values. Here there is no MVs.
Unique shows; that there are several different values in each column, for example, we had three races (white, black, and other), now it shows us the value of 3. Here all the values are correct, as we had previously estimated by Notepad++ and Excel.
Top shows: which of the samples has been repeated the most? For instance, in the first column, as we have already noticed, the white race is repeated the most. The rest of the columns are the same.
Frequency shows: that how many times ***Top*** has been repeated?

What's going on in our Jupyter Notebook?

.dropna()

****** Pleas note that in DataMean we had no Missing Value, because by implementing mean of the DataFrame, Missing Values are gone. ******

6- If we noticed in the previous section that we have MVs in the dataset, in this section, we will delete it with the .dropna() command.
We didn't have any missing values here, so the .dropna() has no effect on this DataFrame.

```
1 DF = DataMean.dropna()
2 DF
```

	Date	Temperature (C)	Wind Speed (km/h)	Pressure (millibars)	Humidity
0	2006-01-01	3.873148	21.372750	1012.279167	0.818333
1	2006-01-02	5.418519	17.551683	1010.131667	0.844583
2	2006-01-03	2.319444	8.417617	1020.805000	0.898333
3	2006-01-04	2.274074	11.579925	981.826667	0.905417
4	2006-01-05	2.698148	9.515100	935.988333	0.948333
...	...	...	...	...	...
4013	2016-12-27	0.280324	10.980200	1020.304583	0.890000
4014	2016-12-28	0.224306	10.969467	1020.334583	0.890000
4015	2016-12-29	0.169676	10.892992	1020.365833	0.889583
4016	2016-12-30	0.119444	10.806454	1020.395000	0.889167
4017	2016-12-31	0.072454	10.764862	1020.423750	0.888750

4018 rows × 5 columns

```
1 DF.shape
```

```
(4018, 5)
```

What's going on in our Jupyter Notebook?

Plots

7- Then, we implement different plots and graphs, which we will explain in the next section (here is a sample code of scatter plotting).

```
# PLOTTING DF1

plt.figure(figsize=(10, 6), dpi=80) # Maximize the plot

plt.scatter(DF['Temperature (C)'], DF['Humidity'], c='rebeccapurple', alpha=0.5)

plt.legend (['Temperature (C)'], loc = 'best')
plt.title ("England Weather Humidity", fontsize=20)

plt.xlabel('Temperature (C)', fontsize=15)
#plt.xticks(rotation=90) # Rotating the x labels
plt.ylabel('Humidity', fontsize=15)

#plt.grid()

#plt.savefig ('D:/IMT/3- Data Science/5- Project #1/plt.savefig/Scatter plots (NEW)/1-Temperature&Humidity.png')

plt.show()
```

03

Plots

Scatter plot,
Bar graph,
Overlap Scatter plot,
and
Count plot.

What is a Scatter plot?

A scatterplot shows the relationship between two quantitative variables measured for the same individuals.

The values of one variable appear on the horizontal axis, and the values of the other variable appear on the vertical axis.

Each individual in the data appears as a point on the graph.

How to describe a Scatter plot?

1) **Form:** Is the association linear or nonlinear?
2) **Direction:** Is the association positive or negative?
3) **Strength:** Does the association appear to be strong, moderately strong, or weak?
4) **Outliers:** Do there appear to be any data points that are unusually far away from the general pattern?

What is a Bar graph?

A bar graph is a nice way to display categorical data.

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

The bars can be plotted vertically or horizontally. A vertical bar chart is sometimes called a column chart.

How to describe a Bar graph?

1. In the first paragraph, give basic details about the chart including what it shows, where it refers to and when.
2. When you describe chart data, be specific. Mention the category and figure.
3. A trend is a change over time. To describe trends, focus on what is increasing or decreasing compared to some time in the past.
4. If several categories show the same trend, talk about them together.
5. State the units of measurement.
6. Many of the verbs for up and down trends can also be used as nouns.

What is a Count plot?

Show the counts of observations in each categorical bin using bars.

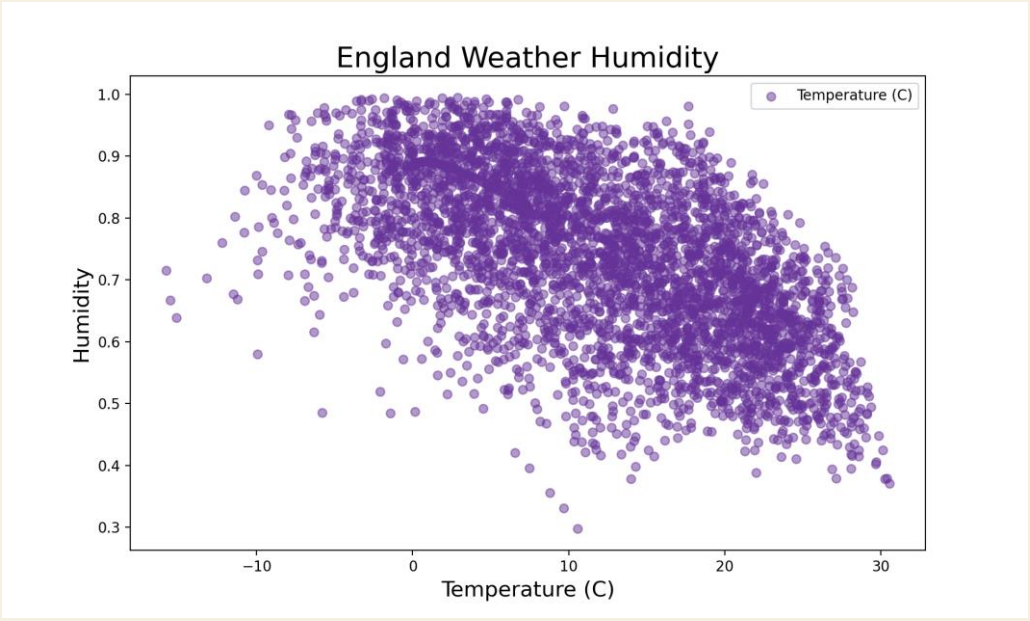
A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable.

To answer what is the difference between Barplot and Countplot in Seaborn, we have to say that, countplot plots the count of the number of records by category while barplot plots a value or metric for each category (by default, barplot plots the mean of a variable, by category)

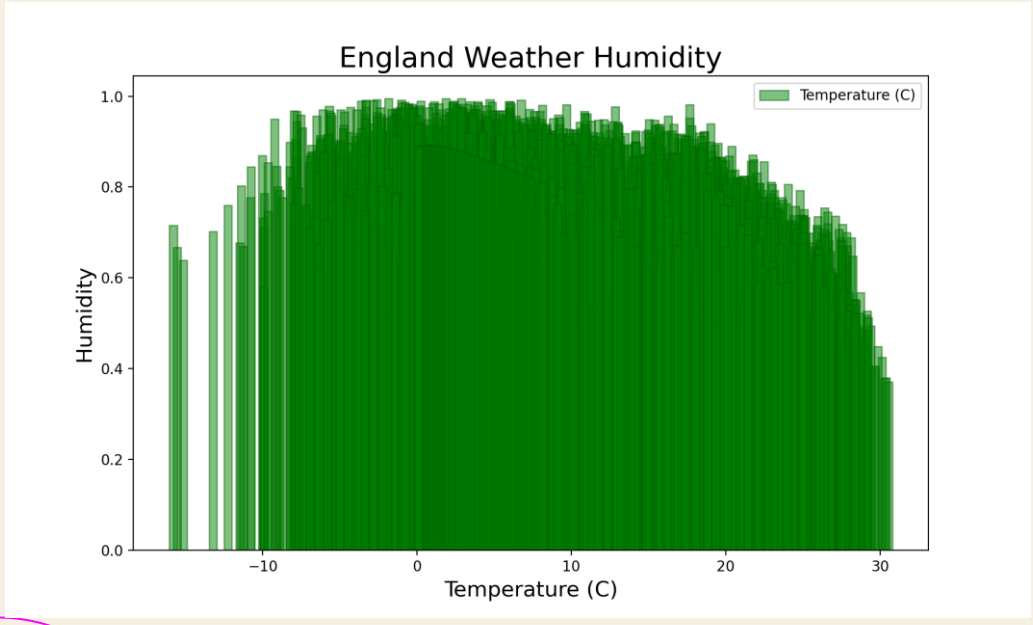
How to describe a Count plot?

1. The countplot is used to represent the occurrence (counts) of the observation present in the categorical variable.
2. It uses the concept of a bar chart for the visual depiction.

Scatter Plot #1
Temperature vs. Humidity



Bar Plot #1
Temperature vs. Humidity

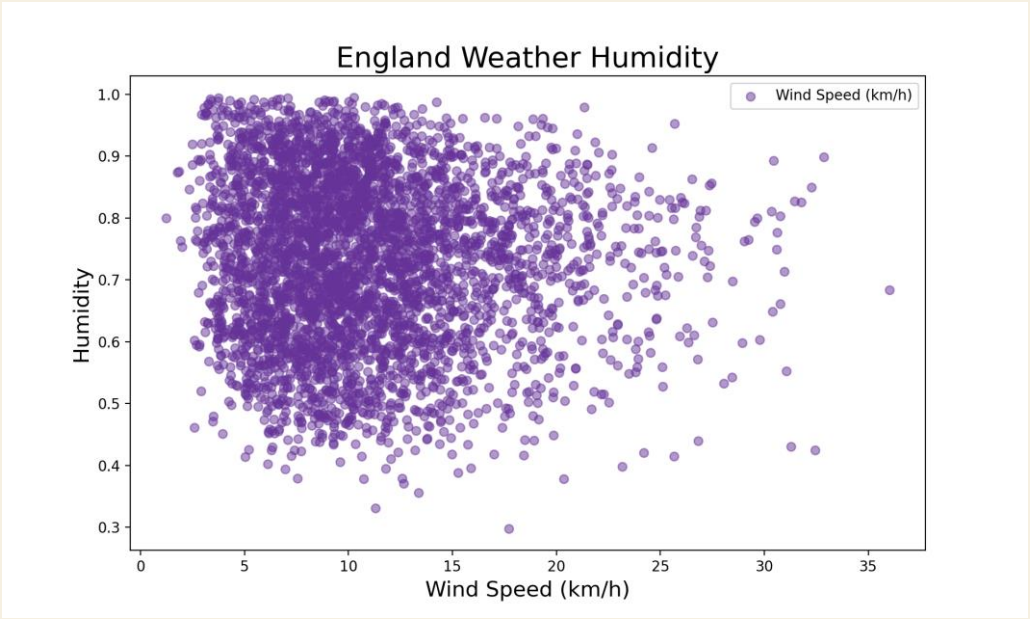


Plot #1 description

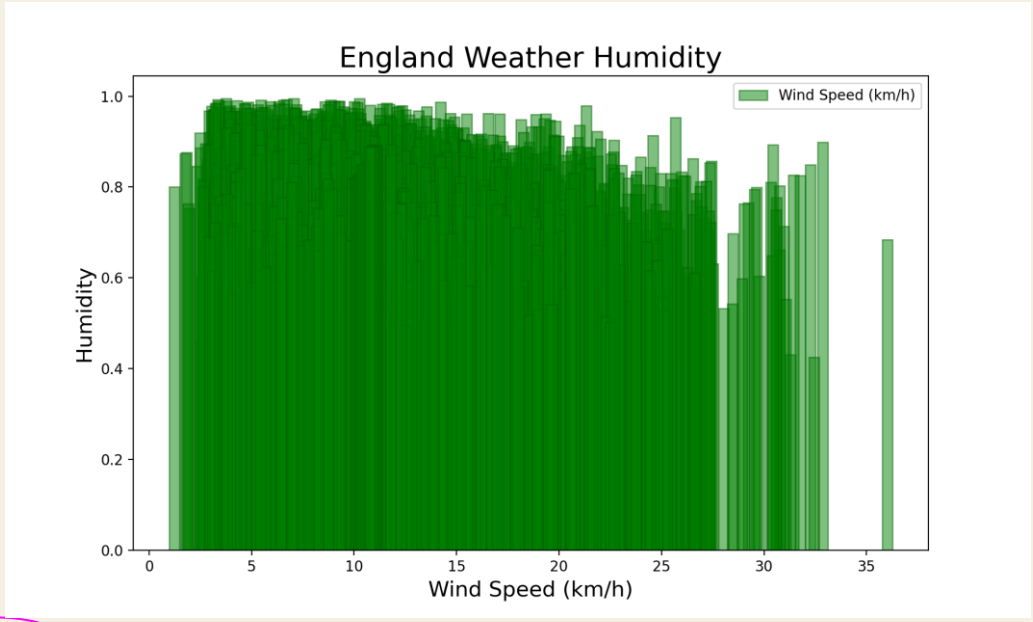
The Scatter plot:
The chart shows the temperature in °C which is changing from -20 to +30 on the x-axis. It also shows the humidity change from 0.3 to 1 on the y-axis. This scatter plot shows a strong, negative, linear association between the temperature and the humidity with a few potential outliers.

The Bar graph:
The graph shows the temperature in °C which is changing from -20 to +30 on the x-axis. It also shows the humidity change from 0 to 1 on the y-axis. Temperature between -8 to 18 °C has the most humidity value, which is close to 1. The graph shows that as the temperature increases from minus degrees to 0 degrees, we also have an increase in humidity. From 0 degrees to about 20 degrees, the humidity is at its maximum, which is close to 1. As the temperature increases from 20 degrees to more than 30 degrees, the humidity fell dramatically.

Scatter Plot #2
Wind Speed vs. Humidity



Bar Plot #2
Wind Speed vs. Humidity

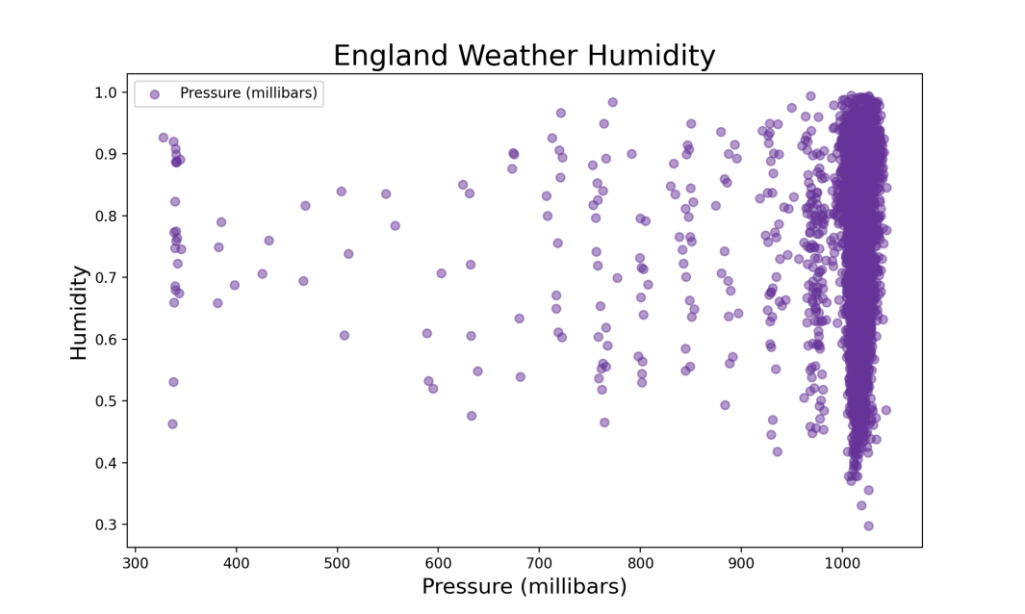


Plot #2 description

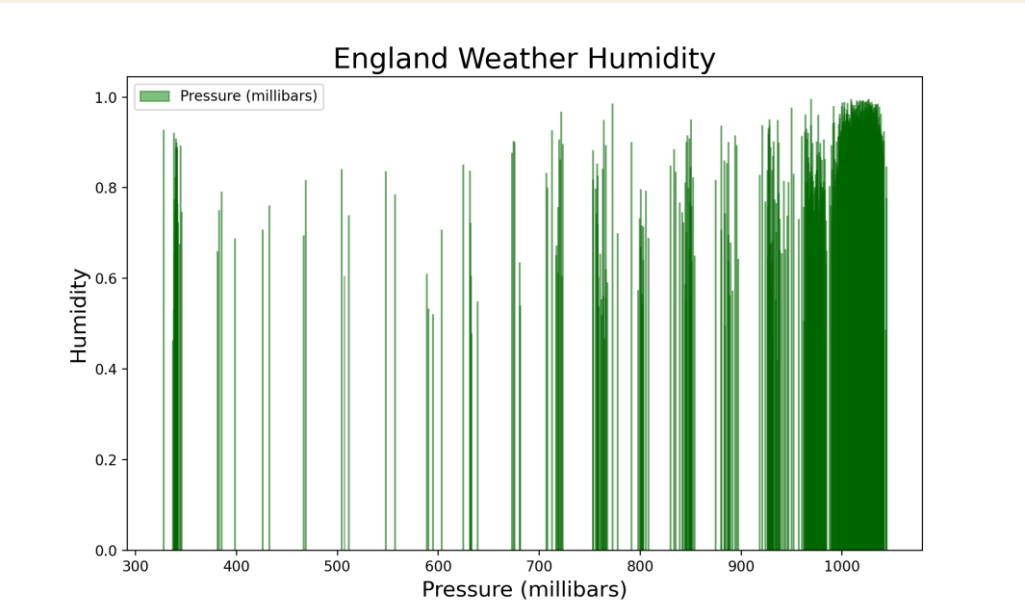
The Scatter plot:
The chart shows the wind speed in km/h which is changing from 0 to 35 on the x-axis. It also shows the humidity change from 0.3 to 1 on the y-axis. This scatter plot shows a strong association between the wind speed and the humidity with a few potential outliers.

The Bar graph:
The graph shows the wind speed in km/h which is changing from 0 to 35 on the x-axis. It also shows the humidity change from 0 to 1 on the y-axis. Wind speed between 2 to 12 km/h has the most humidity value, which is close to 1. The graph shows that as the wind speed increases from 12 to 27 km/h, we have a decrease in humidity. But as the wind speed increases from 27 to 33 km/h, humidity goes up.

Scatter Plot #3
Pressure vs. Humidity



Bar Plot #3
Pressure vs. Humidity



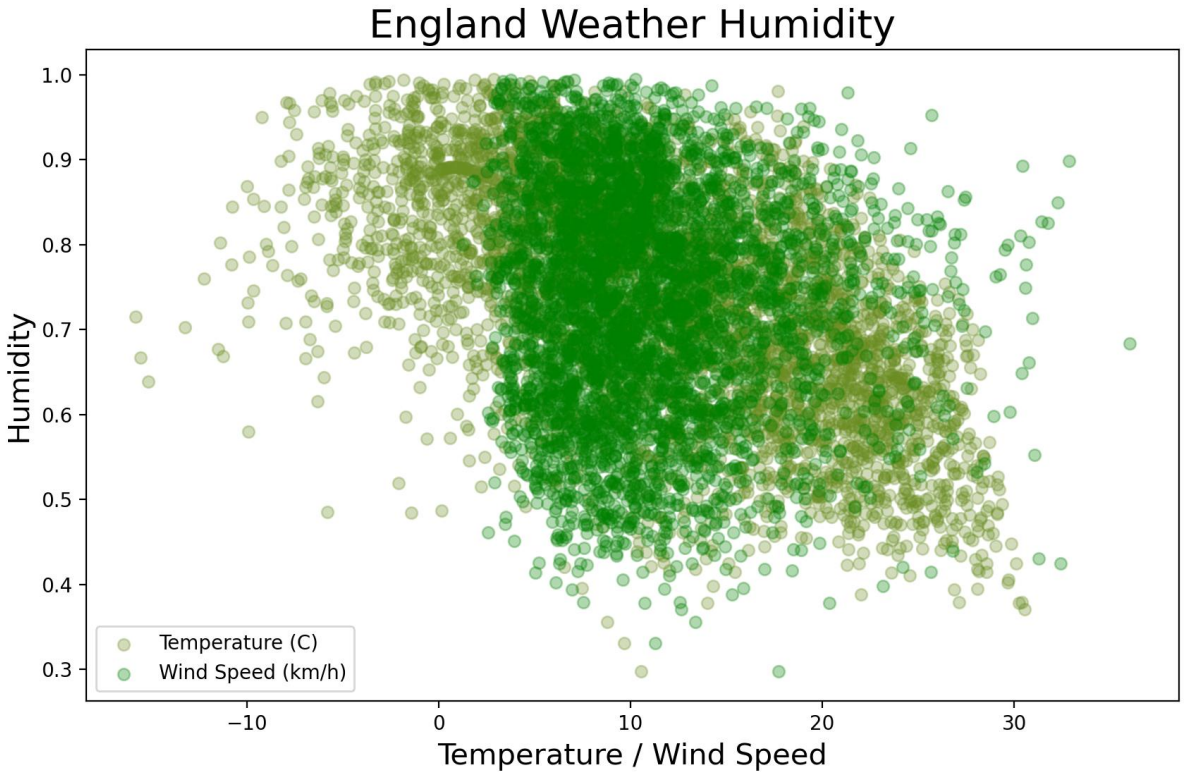
Plot #3 description

The Scatter plot:
The chart shows the pressure in millibars which is changing from 300 to 1000 on the x-axis. It also shows the humidity change from 0.3 to 1 on the y-axis. This scatter plot shows a strong, linear association between the pressure and the humidity. There don't appear to be any outliers in the data.

The Bar graph:
The graph shows the pressure in millibars which is changing from 300 to 1000 on the x-axis. It also shows the humidity change from 0 to 1 on the y-axis. Pressure between 330 to 350 millibars & pressure between 700 to 780 millibars & pressure between 980 to 1150 millibars has the most humidity value, which is close to 1. The graph shows that as the pressure increases, we have repeated increasing decreasing values in humidity.

Over lap Scatter Plot

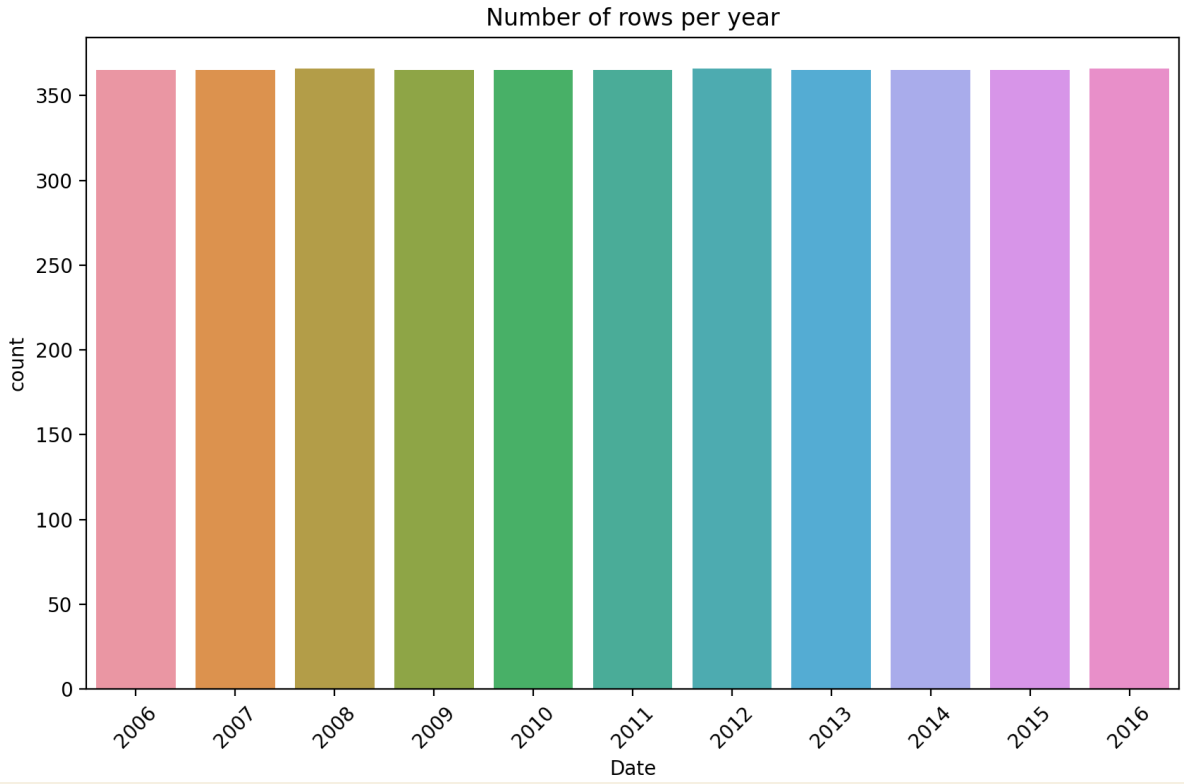
Temperature / Wind Speed vs. Humidity



The Scatter plot:

The chart shows the temperature in °C and the wind speed in km/h which are changing from -20 to +30 on the x-axis. It also shows the humidity change from 0.3 to 1 on the y-axis. This scatter plot shows a strong, almost negative, almost linear association between the temperature & wind and the humidity with a few potential outliers. This overlap scatter also indicates that the temperature data points are more sparse and distributed than the wind speed data points at the same humidity levels. As the temperature trend is decreasing, the wind speed trend is more focused in the middle of the graph.

Count Plot
By Seaborn



The Count plot:

As mentioned earlier, this graph is used to show the occurrence (count) of an observation in a categorical variable.

On the x-axis, you can see the years of registration of this dataset, which shows from 2006 to 2016, and it is for 11 consecutive years.

On the y-axis, you can see the number of counts for each year. This plot reports more than 350 cases for each year.

04

Conclusion

What was done.

What was achieved.

Summary & Conclusion

In this report, the England Weather dataset was examined, necessary pre-processing was done and plots were drawn.
Plots were based on the comparison of Feature and Target.
Each of these plots showed us relationships between samples, data points, and data within the dataset. Relationships that were not possible to discover in normal mode and only by looking at the table called dataset. These relationships give the reader of the report, even if he/she does not have expertise and knowledge about this dataset, valuable and categorized information.
The purpose of plotting these data is to discover the relationships between the data.

References

For describing Histograms:

Histograms review (article) | Khan Academy

For describing Scatter plot:

Describing scatterplots (form, direction, strength, outliers) (article) | Khan Academy

For describing Bar graph:

Bar graphs review (article) | Khan Academy

The end

Do you have any questions?

Samira.Shemirani92@gmail.com

www.linkedin.com/in/samira-shemirani-664302132