CHANDIGARH GROUP OF COLLEGES

Building Careers. Transforming Lives.

Six weeks industrial training report

(5th semester)

On

ANALYSIS OF WEATHER DATASET

Submitted for the partial fulfilment of degree

Of

Bachelor of Technology (Artificial

Intelligence & Data Science)

Submitted by:

Anshul Sharma

University Roll no: - 2121704

submitted to:

DECLARATION

I Anshul Sharma student of B. TECH (Artificial Intelligence & Data Science) 5th semester studying at **Chandigarh group of college Landran** hereby declare that the **six-week Industrial training Report,** undergone at **"Solitaire Infosys"** submitted to **Punjab Technical University, Kapurthala** impartial fulfillment of the award of degree of Bachelorof Technology in Artificial Intelligence & Data Science is the original work conducted by me.

The information and the data in the report is authentic to the best of my knowledge.

Anshul Sharma

ACKNOWLEDGEMENT

I student of Chandigarh group of college Landran have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to the "Solitaire Infosys" for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

I would like to express my gratitude towards my teachers for their kind co-operation and encouragement which help me in completion of this project.

Anshul Sharma

Certificate

TABLE OF CONTENTS

INTRODUCTION

Accurate weather information plays a vital role in shaping both individual decisions and organizational strategies. Businesses across various sectors heavily rely on weather conditions to optimize operations and plan effectively.

✓ Importance of Weather data:

Weather data serves as a cornerstone for individuals and organizations alike. Businesses, agriculture, transportation, and outdoor events all hinge on weather conditions. Accurate weather data aids in making informed decisions, mitigating risks, and improving resource allocation. With the ease of accessing historical meteorological data online, insights can be derived to understand trends and patterns, making it a valuable resource for planning and adaptation

.

• Overview of Exploratory Data Analysis (EDA):

Exploratory Data Analysis is an approach to analyze data, to summarize the main characteristics of data, and better understand the data set. It also allows us to quickly interpret the data and adjust different variables to see their effect. The three main steps to get a perfect EDA are *extracting* the data from an authorized source, *cleaning* and *processing* the data, and performing *data visualization* on the cleaned data set.

• Project Objective:

The primary objective of the project was to analyze the influence of global warming on weather parameters. Specifically, the hypothesis focused on the increase in apparent temperature and humidity over a 10-year period from 2006 to 2016.

Hypothesis:

A hypothesis is an assumption, an idea that is proposed for the sake of argument so that it can be tested to see if it might be true.

- Null Hypothesis:

The Null Hypothesis H0 is *"Has the Apparent temperature and humidity compared monthly across 10 years of the data indicate an increase due to Global warming"* — That means we need to find whether the average Apparent temperature for the month of a month says April starting from 2006 to 2016 and the average humidity for the same period have increased or not.

Terminologies:

Meteorological Data refers to data consisting of physical parameters that are measured directly by instrumentation, and include temperature, dew point, wind direction, wind speed, cloud cover, cloud layer(s), ceiling height, visibility, current weather, and precipitation amount.

Apparent temperature is the temperature equivalent perceived by humans, caused by the combined effects of air temperature, relative humidity, and wind speed. The measure is most commonly applied to the perceived outdoor temperature.

Humidity is the amount of water vapor in the air. If there is a lot of water vapor in the air, the humidity will be high. The higher the humidity, the wetter it feels outside.

Dataset:

The dataset currently using, can be obtained from Kaggle. The dataset has hourly temperature recorded for the last 10 years starting from *2006–04–01 00:00:00.000 +0200* to *2016–09–09 23:00:00.000 +0200*. It corresponds to Finland, a country in Northern Europe.

URL: https://www.kaggle.com/datasets/muthuj7/weather-dataset

Data Preprocessing:

Before analysis, data preprocessing is essential to ensure accuracy and consistency. Anaconda Environment in conjunction with Visual Studio Code was used for this purpose. The initial steps involved importing essential Python libraries, loading the dataset, and exploring its basic properties. The dimensions, data types, missing values, and distinct elements were investigated to gain a comprehensive overview of the dataset.

Python libraries imported:

- **Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- **NumPy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **Matplotlib:** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.
- **Seaborn:** Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.

Different function:

- **read_csv():**

Load the dataset as the dataset is in CSV form.

- **head():**

read the first 5 rows from data.

- **.shape:**

tells the total number of rows and column.

- **.info():**

To find out the types and overall summary of the data frame. It comes in handy when doing exploratory analysis of the data.

- **describe():**

We can use the **describe()** function to get the descriptive statistical details of the data-frame.

- **nunique():**

To check the distinct elements in the data frame.

- **isnull():**

it check for any missing values in data.

- **isnull().sum():**

to get the total number of missing value in each columns.

Correlation Analysis:

Correlation analysis refers to the process of examining the relationships between different columns (variables) in the weather dataset. It aims to quantify how changes in one variable are associated with changes in another variable, specifically focusing on the strength and direction of these relationships. The correlation analysis is done using the Pandas library in Python and is visualized using a heatmap created with Seaborn**.**

The correlation analysis involves the following parts:

Calculating Correlation Matrix: The code calculates the correlation coefficients between all pairs of columns in the dataset. The resulting matrix, known as the correlation matrix, contains correlation coefficients for each combination of variables. This is achieved using the data.corr() function.

Creating a Correlation Heatmap: The correlation matrix is visualized as a heatmap using Seaborn's sns.heatmap() function. In the heatmap, each cell corresponds to the correlation between two variables, and the color intensity represents the strength of the correlation. Brighter colors indicate stronger correlations.

Interpretation of Correlation Heatmap: By observing the heatmap, you can identify which pairs of variables have stronger correlations. Positive correlations are indicated by colors moving towards the brighter side, while negative correlations are indicated by colors moving towards the darker side. Variables with higher positive correlation coefficients are likely to increase together, and variables with higher negative correlation coefficients are likely to have one increase while the other decreases.

Data Parsing and Resampling:

Data parsing and resampling are used to convert the original hourly weather data into a more manageable and meaningful format for analysis. This process involves transforming the data from hourly measurements to monthly averages. The steps for data parsing and resampling are as follows:

Data Parsing:

Converting 'Formatted Date' to Datetime: The code converts the 'Formatted Date' column in the dataset to a datetime format using Pandas' pd.to_datetime() function. This allows for easier manipulation and indexing of the date-time values.

Setting 'Formatted Date' as the Index: After converting the 'Formatted Date' column, it is set as the index of the DataFrame using the data.set_index("Formatted Date") function. This indexing facilitates chronological data analysis.

Data Resampling:

Resampling is a technique to change the frequency of the data while maintaining the data's integrity and capturing important patterns. In this case, the hourly data is resampled to a monthly frequency.

Selecting Relevant Columns for Hypothesis Testing: The code selects two columns, namely 'Apparent Temperature (C)' and 'Humidity', as they are relevant for testing the hypothesis about the influence of global warming on these variables.

Resampling Using Mean: The selected columns are then resampled using the resample() function with the argument "MS" (Month Start) to specify the resampling frequency as the start of each month. The .mean() function is applied to calculate the average values for each month.

The process of data parsing and resampling helps in simplifying the dataset and converting it into a more meaningful format for analysis.

Relation between Apparent Temperature & Humidity:

The relationship between apparent temperature and humidity is explored using a regression plot. This analysis aims to visualize and understand the connection between these two meteorological parameters.
Let's break down the components of this analysis:

Relation between Apparent Temperature & Humidity:

Using Regression Plot (regplot) for Visualization: A regression plot is created using Seaborn's sns.regplot() function. This plot helps visualize the relationship between two variables and fit a regression line to the data points. In this case, the x-axis represents the "Apparent Temperature (C)" values, and the y-axis represents "Humidity" values.

Interpretation of Linear Relationship: The regression line's slope and direction provide insights into the linear relationship between apparent temperature and humidity. A negative slope indicates an inverse correlation: as apparent temperature increases, humidity tends to decrease, and vice versa. The scattered data points around the regression line show the variability in the relationship.

Parts of the Analysis:

Calling regplot function and Assigning Data: The code calls sns.regplot() and assigns the data to be plotted. The "Apparent Temperature (C)" column is used as the x-axis data, and the "Humidity" column is used as the y-axis data.

Color Parameter: The color parameter "color" is specified as "r" to set the color of the plotted regression line.

Title for the Plot: The plt.title() function sets a title for the plot, such as "Relation between Apparent Temperature (C) and Humidity."

Saving the Figure: The plt.savefig() function is used to save the plot as an image file (e.g., 'plot2.png') with a specified dpi (dots per inch) value and layout settings.

Displaying the Plot: Finally, plt.show() is used to display the regression plot on the screen.

This analysis visually presents the relationship between apparent temperature and humidity. The observed negative slope indicates that as the apparent temperature changes, there is a corresponding change in humidity in the opposite direction. This relationship aligns with the common understanding that air can hold more moisture at higher temperatures, leading to lower humidity levels

Yearly Variation of Apparent Temperature and Humidity:

The yearly variation of apparent temperature and humidity is examined using a line plot. This analysis aims to visualize how apparent temperature and humidity change over the years.

Using Line Plot (lineplot) to Visualize Variation over Time: A line plot is created using Seaborn's sns.lineplot() function. This plot helps visualize the trends and patterns in data over time. In this case, the x-axis represents the years (time), and the y-axis represents the average values of "Apparent Temperature (C)" and "Humidity."

Observations from Yearly Variation Plot: The analysis aims to observe trends in the yearly variation of apparent temperature and humidity. The plot provides insights into how these two meteorological parameters have changed over the years.

Parts of the Analysis:

Creating Line Plot: The code uses sns.lineplot() to create the line plot. The data being plotted is the df_monthly_mean DataFrame, which contains the average apparent temperature and humidity values resampled by month.

Setting Figure Size: The plt.figure(figsize=(15, 7)) function is used to set the dimensions of the figure for the line plot.

Setting Title and Labels: The plt.title() function sets the title of the plot (e.g., "Variation of Apparent Temperature and Humidity with Time"). The plt.xlabel() function specifies the label for the x-axis ("year").

Saving the Figure: The plt.savefig() function is used to save the plot as an image file (e.g., 'plot3.png') with a specified dpi (dots per inch) value and layout settings.

Displaying the Plot: Finally, plt.show() is used to display the line plot on the screen.

This analysis visually depicts the average apparent temperature and humidity variations over the years from 2006 to 2016. Observing the line plot, you can infer how these meteorological parameters have changed over time. The constancy of humidity contrasted with the fluctuating apparent temperature provides insights into the changing climate conditions and seasonal variations.

Variation of Humidity & Apparent Temperature for All Months:

The variation of humidity and apparent temperature for all months is visualized using a line plot. This analysis aims to observe how humidity and apparent temperature change across different months over the 10-year period.

Defining Month Labeling Function: A function called label_color(month) is defined to assign labels and colors to each month for better visualization. This function maps the month number to the corresponding month name and a color code.

Creating Seaborn Plot to Show Variation by Month: A function named sns_plot(title, data) is defined to create a Seaborn line plot for the variation of humidity and apparent temperature across all months. It iterates through each month, creates a line plot for the corresponding data, and labels it with the month name and color.

Interpretation of Month-wise Variation Plot: The created plot helps in understanding how the average apparent temperature and humidity values change for each month over the 10-year period.

Parts of the Analysis:

Defining Month Labeling Function (label_color(month)): This function takes the month number as input and returns a tuple containing the corresponding month name and color code.

Defining sns_plot(title, data) Function: This function takes a title for the plot and the data (either apparent temperature or humidity) as inputs. Inside the function, a loop iterates through each month (from 1 to 12) and calls another function called plot_month(month, data).

plot_month(month, data) Function: This function takes a month and the corresponding data as inputs. It labels the line plot with the month name and color using the label_color(month) function and creates a line plot using Seaborn's sns.lineplot().

Setting Figure Size: The plt.figure(figsize=(14, 8)) function sets the dimensions of the figure for the line plots.

Setting Title and Labels: The plt.title() function sets the title of the plot (e.g., "Month-Wise Plot for Apparent Temperature of 10 Years"). The plt.xlabel() function specifies the label for the x-axis ("YEAR").

Saving the Figure: The plt.savefig() function is used to save each month-wise plot as an image file (e.g., 'month1.png', 'month2.png', etc.) with a specified dpi (dots per inch) value and layout settings.

Displaying the Plots: Finally, plt.show() is used to display each month-wise plot on the screen.

This analysis provides insights into the variation of humidity and apparent temperature for each individual month over the 10-year period. By examining the series of month-wise plots, you can understand how these meteorological parameters change seasonally and potentially observe any trends or patterns that emerge over time

Snapshot of project:

Source code:

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

```
data = pd.read_csv('weatherHistory.csv')
data.head()
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

First five rows of the dataframe

```
data.shape
Output:
(96453, 12)
```

```
data.describe()
```

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.0	96453.000000
mean	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	0.0	1003.235956
std	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	0.0	116.969906
min	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
25%	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	0.0	1011.900000
50%	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	0.0	1016.450000
75%	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	0.0	1021.090000
max	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	0.0	1046.380000

Output :the statistical details of the dataframe

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  object
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
dtypes: float64(8), object(4)
memory usage: 8.8+ MB
```

Output of the data.info() function

```
data.describe()
```

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.0	96453.000000
mean	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	0.0	1003.235956
std	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	0.0	116.969906
min	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
25%	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	0.0	1011.900000
50%	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	0.0	1016.450000
75%	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	0.0	1021.090000
max	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	0.0	1046.380000

Output :the statistical details of the dataframe

```
data.nunique()
```

```
Formatted Date               96429
Summary                         27
Precip Type                      2
Temperature (C)               7574
Apparent Temperature (C)      8984
Humidity                        90
Wind Speed (km/h)             2484
Wind Bearing (degrees)         360
Visibility (km)                949
Loud Cover                       1
Pressure (millibars)          4979
Daily Summary                  214
dtype: int64
```

Output: distinct elements in dataframe

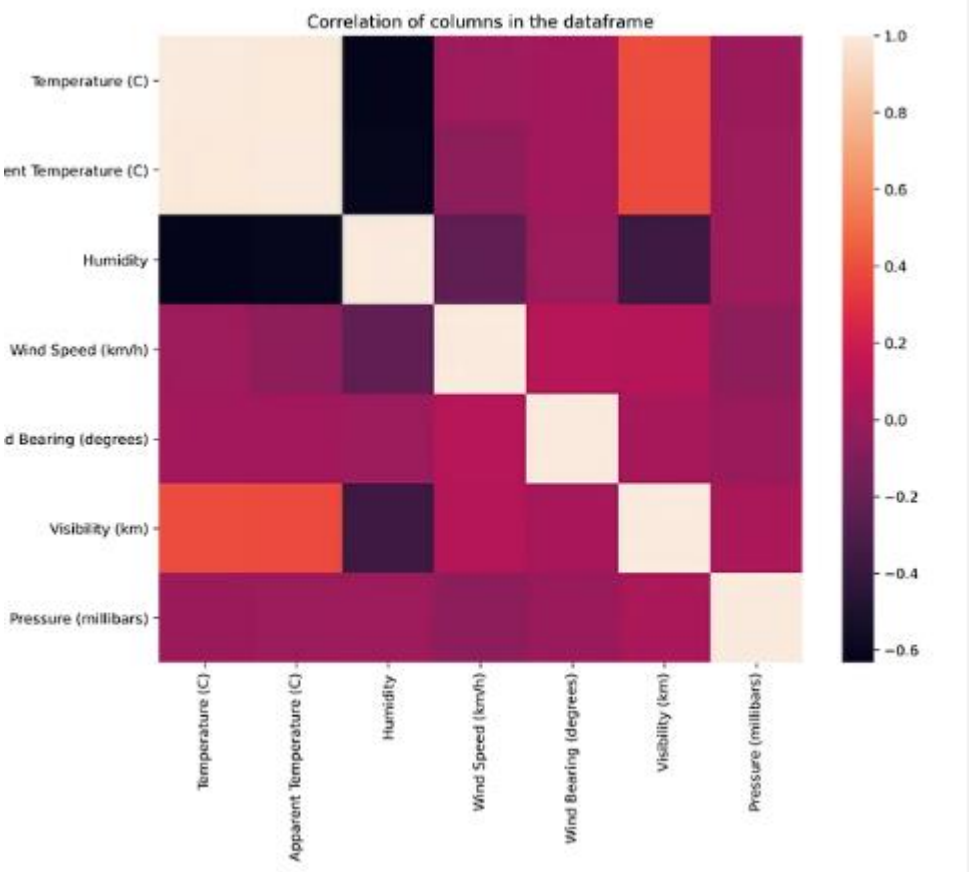
```
data.isnull().sum()
```

```
Formatted Date                 0
Summary                        0
Precip Type                  517
Temperature (C)                0
Apparent Temperature (C)       0
Humidity                       0
Wind Speed (km/h)              0
Wind Bearing (degrees)         0
Visibility (km)                0
Loud Cover                     0
Pressure (millibars)           0
Daily Summary                  0
dtype: int64
```

Total number of missing values in each column

```
data = data.drop(['Loud Cover'], axis = 1)
```

```
# assign data correlation matrix
relation = data.corr()

# Increase the size of the heatmap
plt.figure(figsize=(10,8))

# Store heatmap object in a variable to easily access it when
you want to include more features and you can set the
annotation parameter to True to display the correlation
values on the heatmap.
sns.heatmap(data=relation)

# Give a title to the heatmap.
plt.title("Correlation of columns in the dataframe")

# save the figure.
plt.savefig('plot1.png', dpi=300, bbox_inches='tight')
plt.show()
```



```
data['Formatted Date'] = pd.to_datetime(data['Formatted
Date'],
utc=True)
```

```
data = data.set_index("Formatted Date")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  datetime64[ns, UTC]
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Pressure (millibars)      96453 non-null  float64
 10  Daily Summary             96453 non-null  object
dtypes: datetime64[ns, UTC](1), float64(7), object(3)
memory usage: 8.1+ MB
```

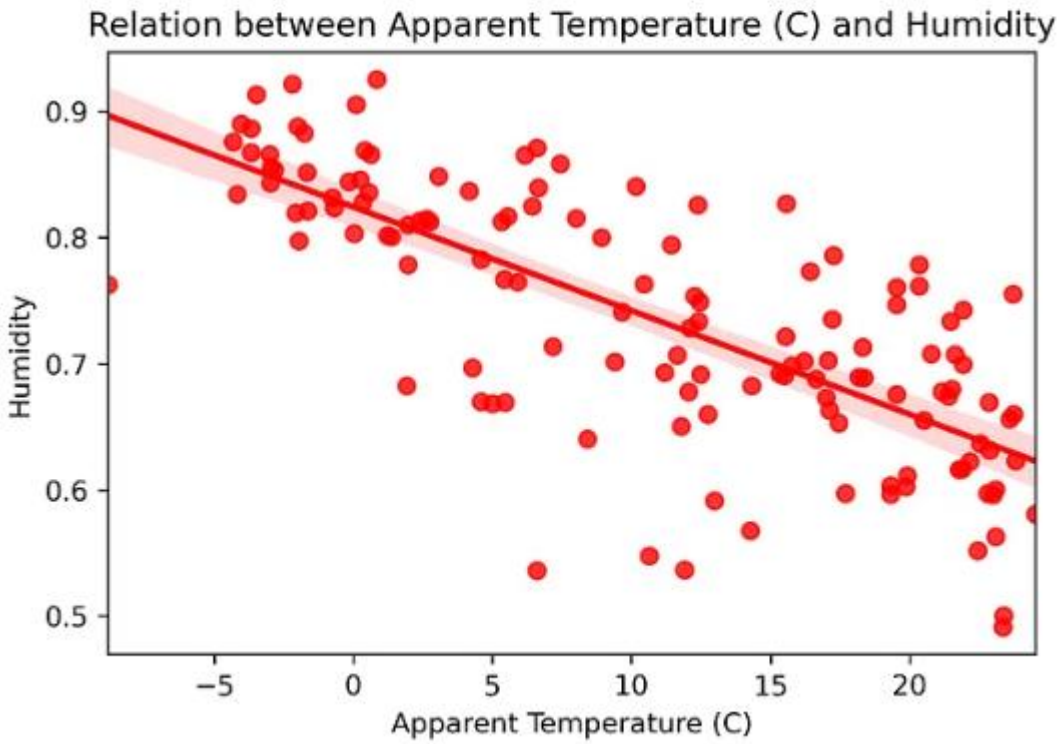
Output : Parsing dates

```
df_column = ['Apparent Temperature (C)', 'Humidity']

df_monthly_mean = data[df_column].resample("MS").mean()
#MS-Month Starting

df_monthly_mean.head()
```

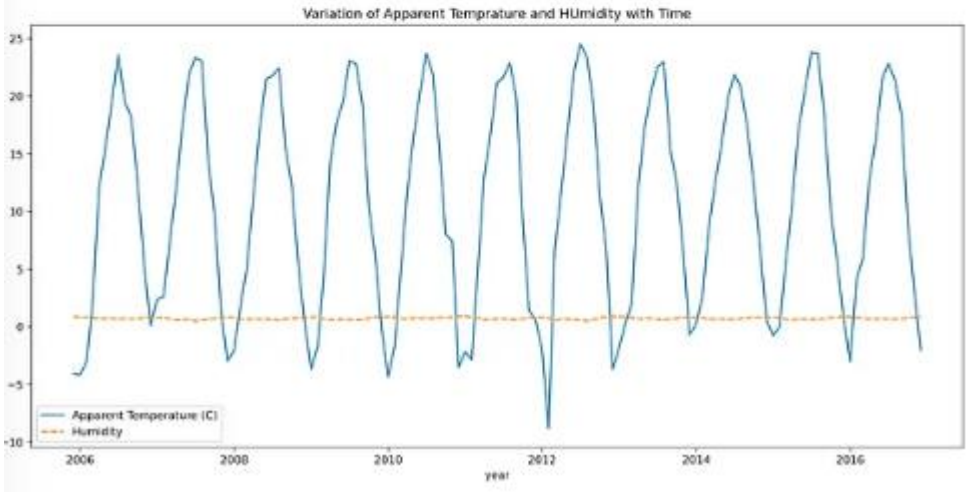
Formatted Date	Apparent Temperature (C)	Humidity
2005-12-01 00:00:00+00:00	-4.050000	0.890000
2006-01-01 00:00:00+00:00	-4.173708	0.834610
2006-02-01 00:00:00+00:00	-2.990716	0.843467
2006-03-01 00:00:00+00:00	1.969780	0.778737
2006-04-01 00:00:00+00:00	12.098827	0.728625

First five rows of Resampled Data

```
# calling regplot function and assign it with our data and plot
labels and color parameter.
sns.regplot(data=df_monthly_mean, x="Apparent Temperature (C)",
y="Humidity", color="r")

# Give a title to the plot
plt.title("Relation between Apparent Temperature (C) and Humidity")

# save the figure
plt.savefig('plot2.png', dpi=300, bbox_inches='tight')
plt.show()
```

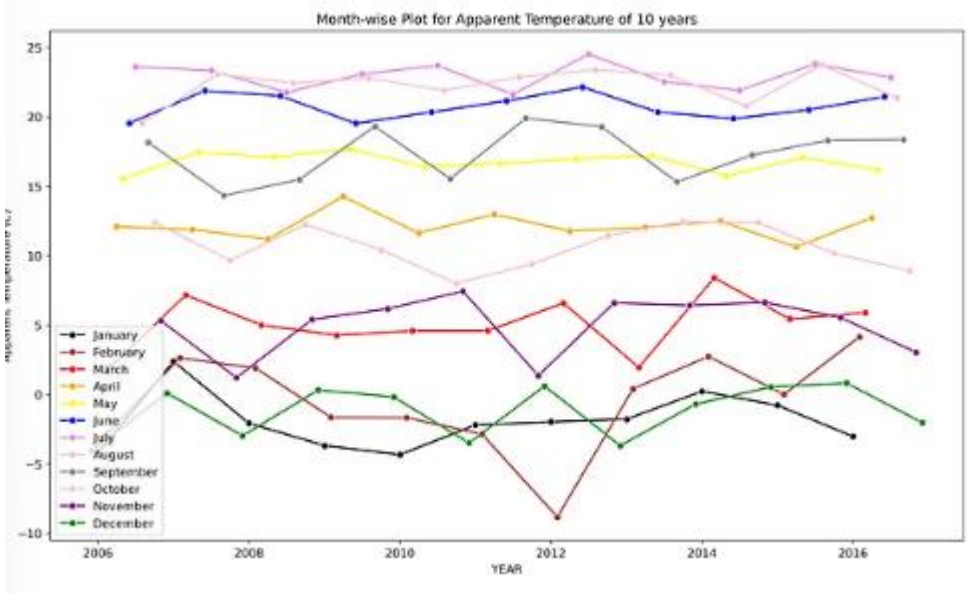


```
plt.figure(figsize=(15,7))
sns.lineplot(data= df_monthly_mean)
plt.xlabel('year')
plt.title('Variation of Apparent Temprature and HUmidity with
Time')
plt.savefig('plot3.png', dpi=300, bbox_inches='tight')
plt.show()
```

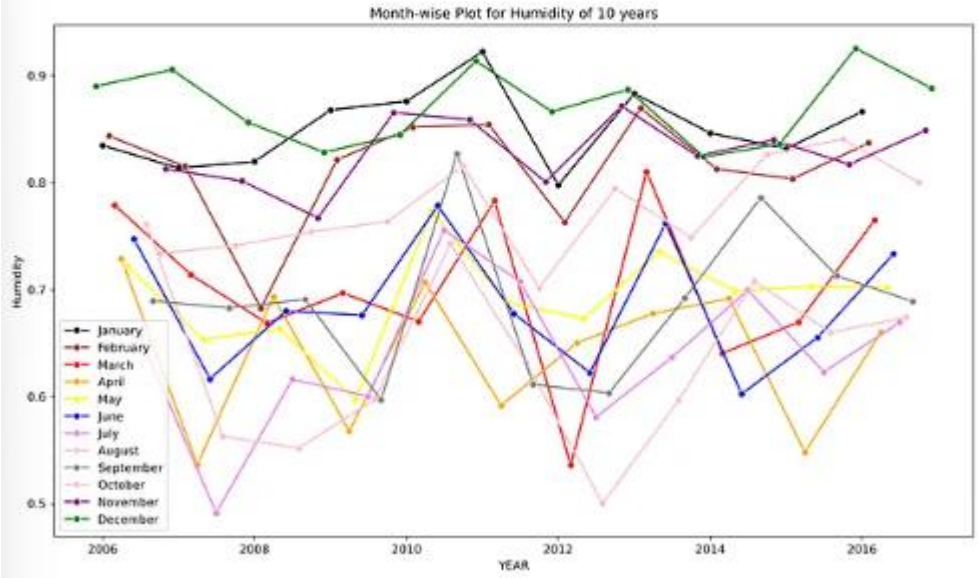



```
# Defining a function call for month to be labeled
def label_color(month):
    if month == 1:
       return 'January','black'
    elif month == 2:
       return 'February','brown'
    elif month == 3:
       return 'March','red'
    elif month == 4:
       return 'April','orange'
    elif month == 5:
       return 'May','yellow'
    elif month == 6:
       return 'June','blue'
    elif month == 7:
       return 'July','violet'
    elif month == 8:
       return 'August','pink'
    elif month == 9:
       return 'September','grey'
    elif month == 10:
       return 'October','pink'
    elif month == 11:
       return 'November','purple'
    else:
       return 'December','green'
```

```
# Assigning variables to resampled data
TEMP_DATA = df_monthly_mean.iloc[:,0]
HUM_DATA = df_monthly_mean.iloc[:,1]

def plot_month(month, data):
    label, color = label_color(month)
    mdata = data[data.index.month == month]

sns.lineplot(data=mdata,label=label,color=color,marker='o')

def sns_plot(title, data):
    plt.figure(figsize=(14,8))
    plt.title(title)
    plt.xlabel('YEAR')
    for i in range(1,13):
        plot_month(i,data)
    plt.savefig('plot4.png', dpi=300, bbox_inches='tight')
    plt.show()

# Month-wise Plot for Apparent Temperature of 10 years
title = 'Month-wise Plot for Apparent Temperature of 10
years'
sns_plot(title, TEMP_DATA)
```

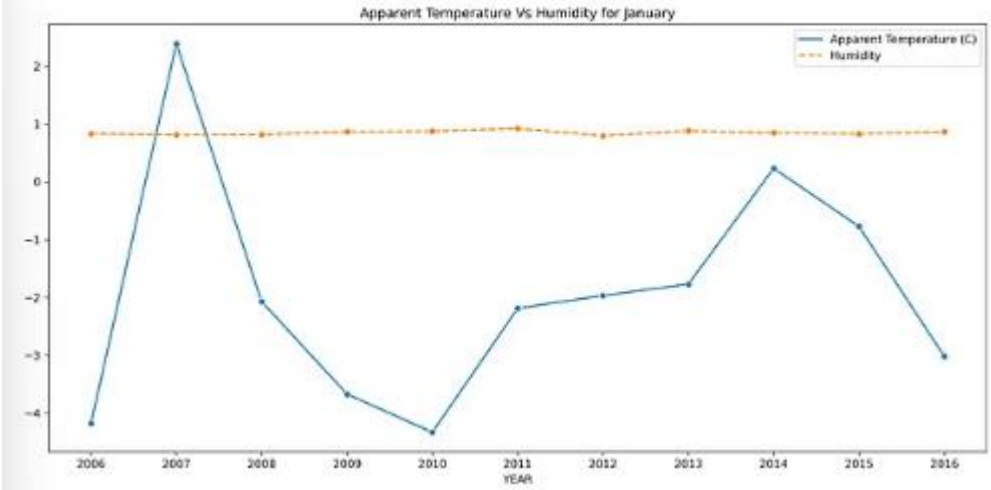


```
# Month-wise Plot for Humidity of 10 years
title = 'Month-wise Plot for Humidity of 10 years'
sns_plot(title, HUM_DATA)
```

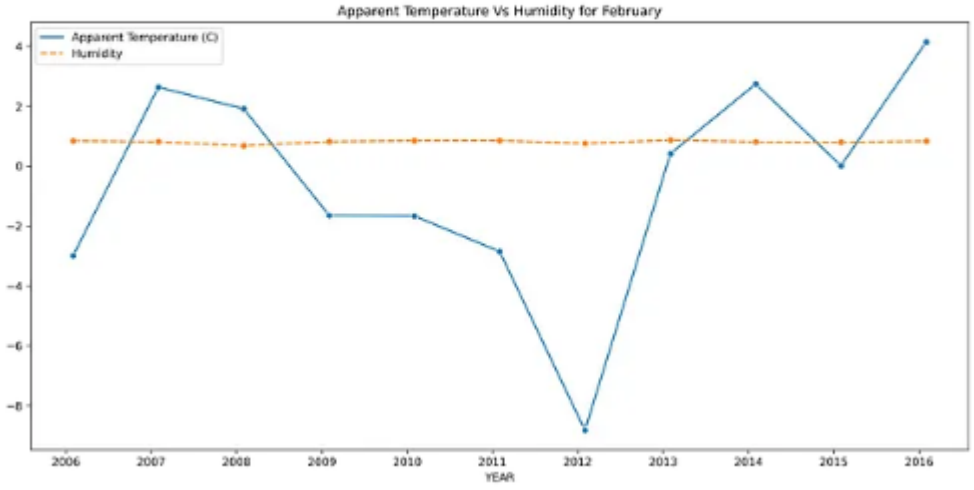



```
# Function for plotting variation for each month
def sns_month_plot(month):
    plt.figure(figsize=(15,7))
    label = label_color(month)[0]
    plt.title('Apparent Temperature Vs Humidity for
{}'.format(label))
    data = df_monthly_mean[df_monthly_mean.index.month ==
month]
    plt.xlabel('YEAR')
    sns.lineplot(data=data, marker='o')
    name="month"+str(month)+".png"
    plt.savefig(name, dpi=300, bbox_inches='tight')
    plt.show()

# Plot for the month of 'January - December'
for month in range(1,13):
    sns_month_plot(month)
```

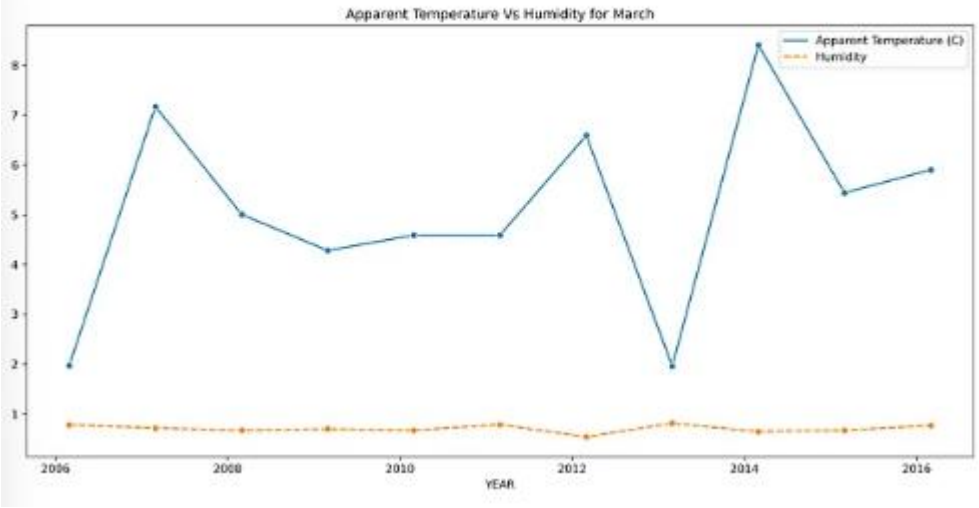
January



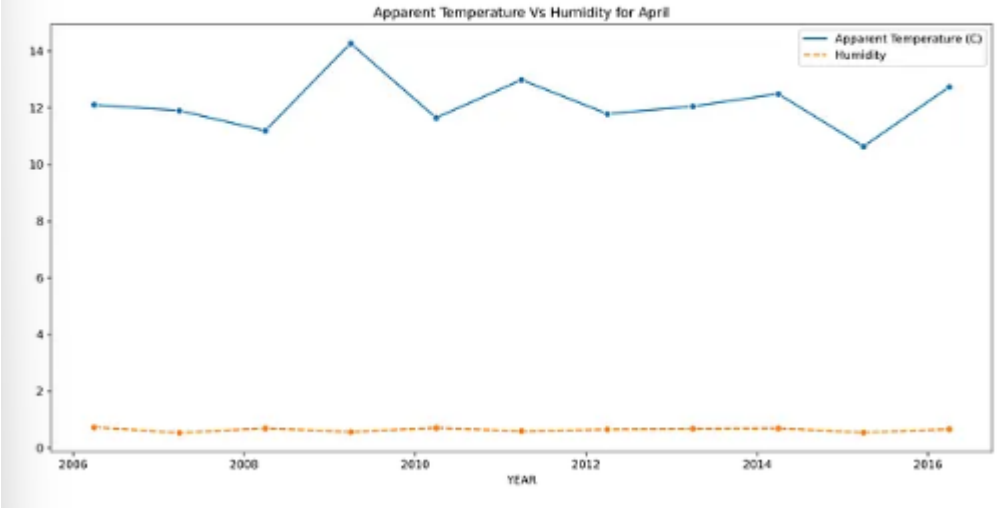
February



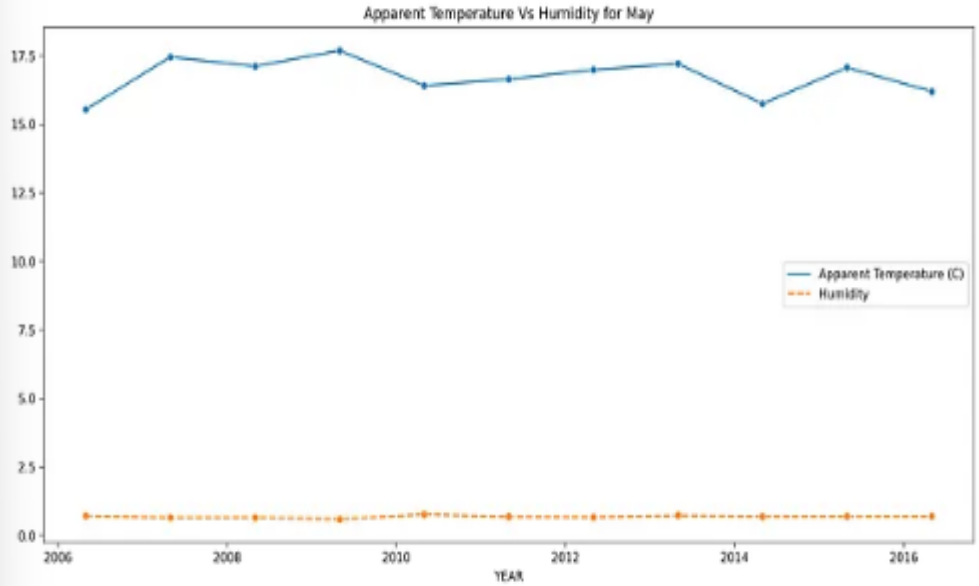
March


Apparent Temperature Vs Humidity for March
Apparent Temperature (C)
Humidity
8
7
6
5
4
3
2
1
2006
2008
2010
2012
2014
2016
YEAR

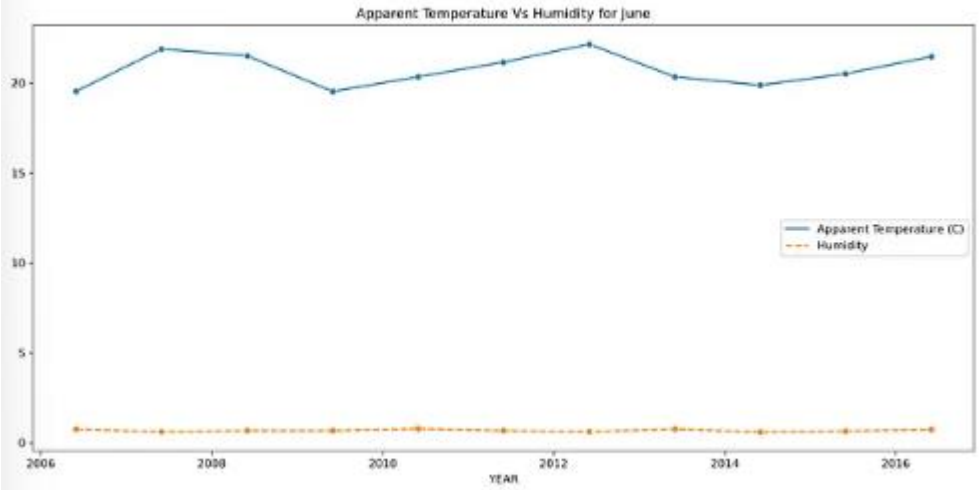
April


Apparent Temperature Vs Humidity for April
Apparent Temperature (C)
Humidity
14
12
10
8
6
4
2
0
2006
2008
2010
2012
2014
2016
YEAR

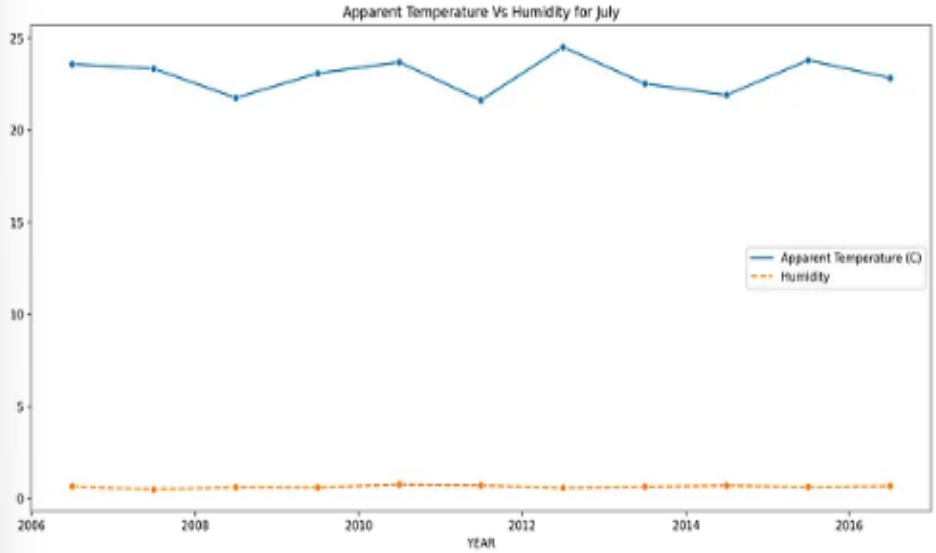
May


Apparent Temperature Vs Humidity for May
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2006
2008
2010
2012
2014
2016
YEAR
Apparent Temperature (C)
Humidity

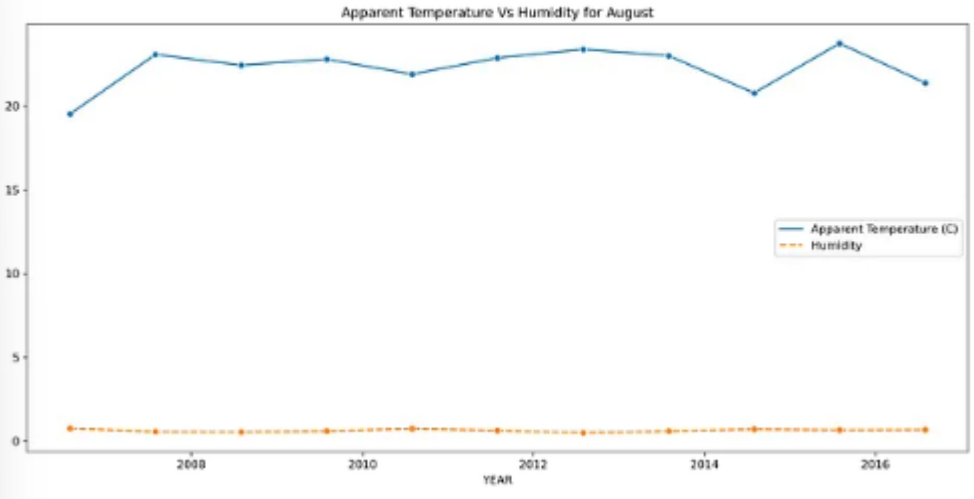
June


Apparent Temperature Vs Humidity for June
20
15
10
5
0
2006
2008
2010
2012
2014
2016
YEAR
Apparent Temperature (C)
Humidity

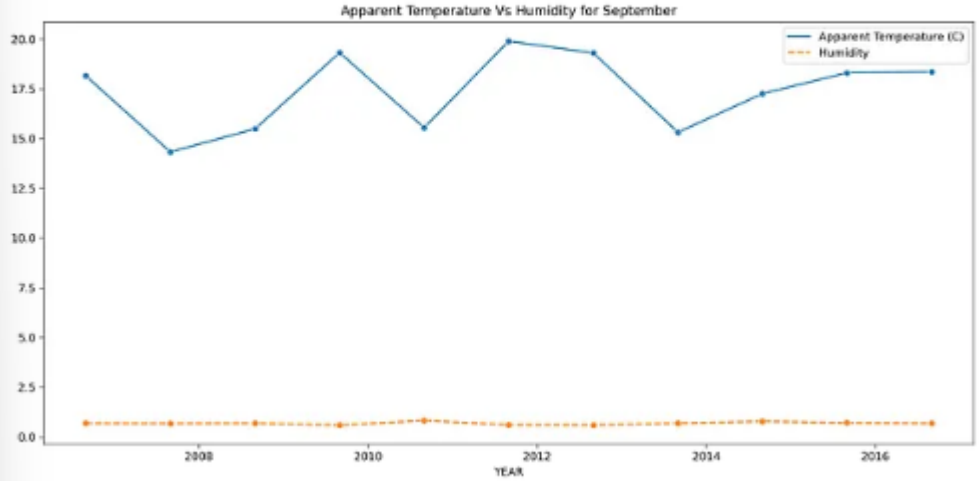
July


Apparent Temperature Vs Humidity for July
25
20
15
10
5
0
2006
2008
2010
2012
2014
2016
YEAR
Apparent Temperature (C)
Humidity

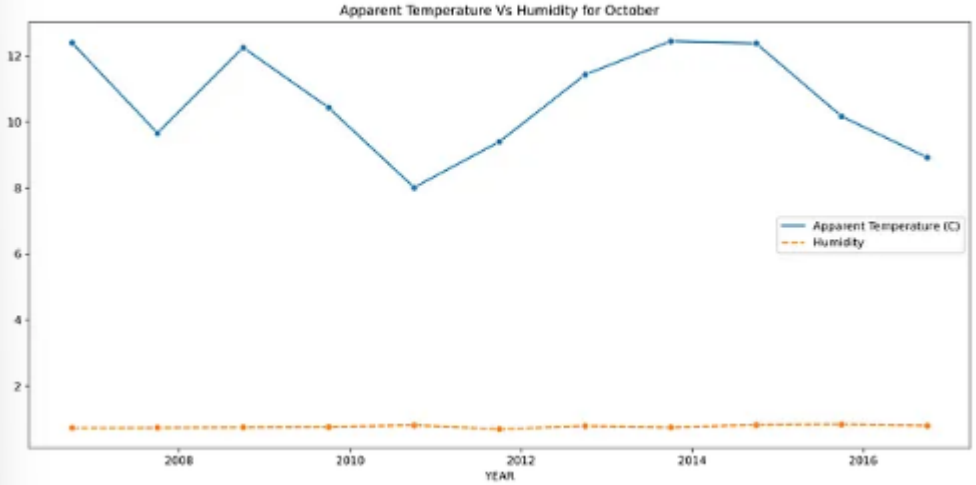
August


Apparent Temperature Vs Humidity for August
20
15
10
5
0
2008
2010
2012
2014
2016
YEAR
Apparent Temperature (C)
Humidity

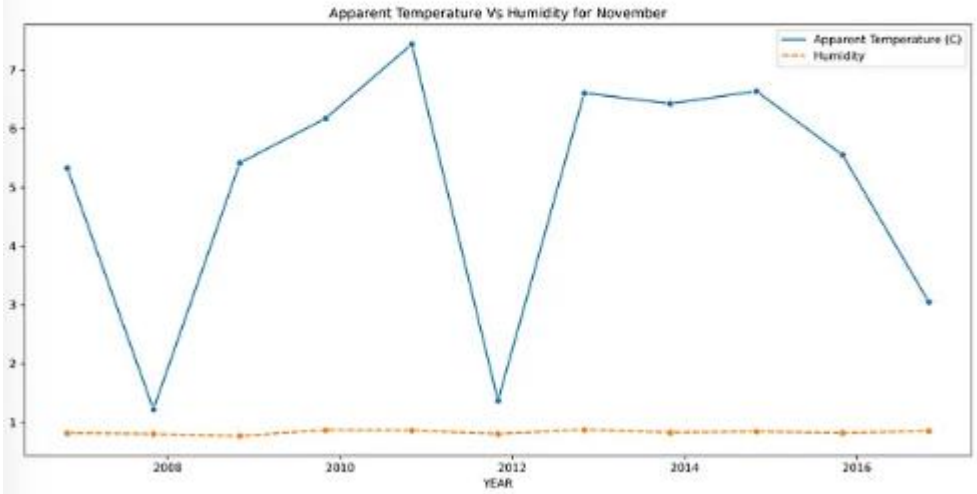
September



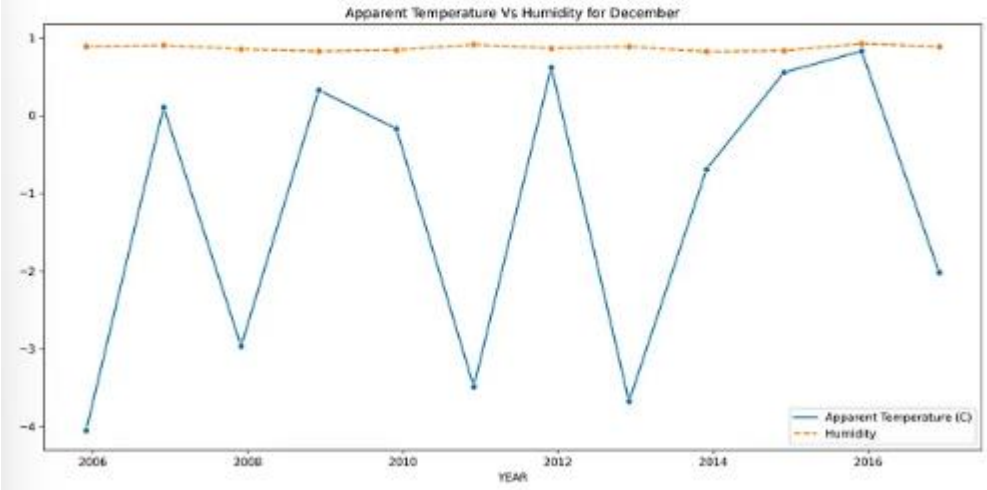
October



November



December



Achievements and Learning:

Achievements:

Data Analysis Proficiency: Successfully conducting EDA demonstrates proficiency in data manipulation, cleaning, visualization, and interpretation. It showcases the ability to handle real-world datasets and derive meaningful insights from them.

Hypothesis Testing: Formulating, testing, and interpreting a hypothesis related to climate change and its impact on weather variables exemplifies the application of statistical methods to real-world problems.

Data Visualization Skills: Creating various types of plots and visualizations helps communicate complex patterns and trends effectively, which is essential for presenting findings to both technical and non-technical audiences.

Project Management: Completing a project from start to finish, including data sourcing, preprocessing, analysis, and reporting, showcases project management skills and the ability to work independently.

Domain Knowledge: Gaining a deeper understanding of meteorological concepts, weather data terminology, and their significance enhances domain-specific knowledge.

Learning Outcomes:

Exploratory Data Analysis: You have learned how to perform exploratory data analysis, which is a foundational skill in data science. EDA helps in understanding data characteristics, identifying patterns, and forming initial hypotheses.

Hypothesis Formulation: You've learned how to formulate a hypothesis and design a study to test it. This process involves critical thinking and the ability to define research questions.

Data Preprocessing: You've gained insights into the importance of data preprocessing, including handling missing values, data type conversions, and dropping irrelevant columns.

Data Visualization: Through creating various plots, you've learned the significance of data visualization in uncovering trends, correlations, and anomalies within datasets.

Statistical Interpretation: You've practiced interpreting descriptive statistics and correlation coefficients to draw meaningful conclusions from data.

Python Programming: Working with libraries like Pandas, NumPy, Matplotlib, and Seaborn has improved your coding skills in Python for data analysis and visualization.

Interdisciplinary Knowledge: You've connected meteorology concepts with data analysis techniques, showing how interdisciplinary knowledge contributes to addressing real-world problems.

Communication Skills: Summarizing your findings and insights in a coherent and structured report improves your ability to communicate technical information clearly.

Critical Thinking: Formulating hypotheses, making observations, and drawing conclusions from data enhance your critical thinking skills, crucial for problem-solving.

Future Research Directions: The project has likely sparked your curiosity about climate-related topics and their broader implications, encouraging you to pursue further research or exploration in related fields.

Overall, this project not only showcases technical skills but also fosters a deeper understanding of the complex relationship between climate variables. It sets a foundation for continuous learning and exploration in data science and climate analysis.

Future Directions:

The exploratory data analysis (EDA) conducted in this project on the weather dataset has provided insights into the relationship between apparent temperature, humidity, and their variations over time. While the EDA is a valuable initial step, it opens up possibilities for more in-depth analysis and future research. Here are potential avenues for further investigation and future findings:

Long-Term Climate Trends: Building upon the observed yearly variation and monthly patterns, further analysis could involve examining longer-term climate trends. This could involve analyzing data from a broader time span or incorporating additional datasets to identify more extensive patterns related to global climate change.

Comparative Analysis: This project focuses on a specific region (Finland). Expanding the analysis to include data from different geographic locations could lead to comparative studies of how global warming's effects vary across regions.

Advanced Statistical Modeling: Employing advanced statistical models, such as time series analysis or machine learning algorithms, could help in predicting future trends in apparent temperature and humidity based on historical data. This could provide insights into potential future climate scenarios.

Climate Extremes and Anomalies: Exploring the dataset for extreme weather events, anomalies, or sudden shifts in temperature and humidity could uncover specific instances of climate-related events and their potential impacts.

Climate Change Mitigation Strategies: Linking the observed climate trends to broader climate change mitigation strategies could provide insights into the effectiveness of different measures taken to address global warming. For instance, analyzing changes in apparent temperature and humidity alongside policy changes, technological advancements, or emissions reductions could be revealing.

Correlation with Other Factors: Investigating the correlation between apparent temperature, humidity, and other factors such as air pollution, urbanization, or land use changes could shed light on the complex interplay of various influences on the climate.

Public Health and Policy Implications: Understanding the relationship between weather patterns and public health outcomes can inform policies and interventions related to heatwaves, cold snaps,

and their impacts on communities.

Predictive Analytics: Using historical data and other relevant features, predictive models could be developed to forecast changes in apparent temperature and humidity in the future. This can have practical applications for preparedness and adaptation.

Feedback Mechanisms: Investigating potential feedback mechanisms, where changes in apparent temperature and humidity might influence other factors like vegetation, ecosystems, or even economic activities, could reveal intricate dynamics.

Climate Resilience Planning: The insights gained from the analysis could contribute to climate resilience planning by identifying vulnerable periods and areas, enabling proactive strategies to mitigate risks.

In summary, the findings from this project serve as a stepping stone for deeper research into the impacts of global warming on weather patterns. The analysis opens up opportunities for multidisciplinary studies that combine climate science, data analytics, and policy considerations to address the complex challenges posed by changing weather conditions.

Conclusion

Summary of Findings and Observations: This part presents a concise overview of the key findings and observations obtained from the entire analysis. It highlights the main insights and trends that were discovered through data exploration.

Conclusion on the Influence of Global Warming on Apparent Temperature and Humidity: This section provides a conclusive statement regarding the project's main objective: testing the influences of global warming on apparent temperature and humidity. Based on the analysis conducted throughout the report, this part addresses whether the hypothesis was supported or not.

Parts of the Conclusion:

Summary of Findings and Observations: This subsection briefly highlights the significant results and patterns identified during the exploratory data analysis. It may include points such as the relationship between apparent temperature and humidity, the variation of these parameters over time and months, and any notable trends.

Conclusion on the Influence of Global Warming: This subsection presents the ultimate outcome of the analysis. It states whether the null hypothesis (H0) was supported or rejected based on the insights gained. If the analysis indicates an increase in apparent temperature and humidity over the years, which aligns with the concept of global warming, the conclusion will affirm the influence of global warming on these meteorological parameters.

By synthesizing the key findings and addressing the hypothesis, the conclusion provides a clear and succinct understanding of the implications of the EDA results. It ties together the various components of the analysis and offers a resolution to the research question posed at the beginning of the project.

Bibliography

- **Data set URL:** https://www.kaggle.com/datasets/muthuj7/weather-dataset
- **DOCUMENTRY URL:** https://sinanthahir.medium.com/experience-analysis-through-weather-data-exploratory-data-analysis-b438ceca8a9b
- **GOOGLE**