School of Computer Science and Engineering

J Component report

Programme : B.Tech

Course Title : Data Visualization

Course Code : CSE3020

Slot : D2

Title: Meteorological Data Analysis and forecasting using ARIMA

GithHub: Akshaykviit023/Meteorological-data-analysis-and-forecasting (github.com)

Team Members:

Akshay Girish | 20BCE1573

Ashwin K S | 20BCE1733

Abstract:

This project aims to show how the Autoregressive Integrated Moving Average (ARIMA) model can be used to analyse meteorological data and forecast future weather patterns. The project is based on an examination of historical weather data from a certain place over a specific time period.

The project starts with a brief overview of ARIMA and its application in time series forecasting. Following that is a full review of the meteorological data used in the study. Temperature, humidity, wind speed, and precipitation data were collected over a number of years. The project then moves on to the data preprocessing procedure, which entails cleaning, normalizing, and converting the data so that it is ready for analysis. The ARIMA model is then applied to the preprocessed data to determine the model parameters that best fit the data. To pick the best ARIMA model, the study applies a variety of statistical approaches, including the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).

The results of the ARIMA model are shown in the following section. The study assesses the ARIMA model's performance using metrics such as Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). The results show that the ARIMA model is quite good at forecasting meteorological data.

Furthermore, the project also investigates the effect of exogenous variables such as seasonal variations, extreme weather occurrences, and climatic changes on the forecasting accuracy of the ARIMA model. According to the study, incorporating these elements considerably enhances the accuracy of the ARIMA model.

In conclusion, this project aims to provides useful insights into the use of ARIMA in meteorological data analysis and forecasting. The project confirms the ARIMA model's accuracy in predicting weather patterns, particularly when exogenous variables are included. The project findings could also be used for weather forecasting and disaster preparedness.

Introduction:

Weather patterns have long played an important role in our lives. Many businesses, including agriculture, transportation, construction, and energy, rely on knowing when and how the weather will change. Advances in technology and computers have resulted in substantial advancements in meteorological data collecting and analysis in recent years. The use of Autoregressive Integrated Moving Average (ARIMA) models for data analysis and forecasting is one such technique that has gained favour in this industry.

ARIMA is a highly effective statistical tool for analysing and forecasting time series data. It was developed in the 1970s by George Box and Gwilym Jenkins and has since become widely used in a variety of industries, including meteorology. ARIMA models can aid in the identification of patterns and trends in weather data, as well as the prediction of future weather conditions based on prior observations. ARIMA models are especially useful when dealing with non-stationarity, which is a typical feature of meteorological data.

The goal of project is to give a thorough examination of the use of ARIMA models in meteorological data processing and forecasting. The project will give a comprehensive understanding of the ARIMA methodology, including its mathematical formulation, assumptions, and limits. The project will employ real-world meteorological datasets to test the usefulness of the ARIMA models in meteorological data processing and forecasting. The paper will show how to fit an ARIMA model to data step by step, covering model selection, parameter estimation, and model diagnostics. The accuracy of the ARIMA model will also be compared to that of other common forecasting techniques, such as exponential smoothing and neural networks, in the study.

This project will cover the practical ramifications of employing ARIMA models in meteorological data processing and forecasting, in addition to the technical components of the research. The project will show the advantages of utilising ARIMA models, such as enhanced accuracy in predicting extreme weather occurrences, increased resource allocation efficiency, and better risk management techniques.

Overall, by proving the usefulness of ARIMA models in predicting weather patterns, this project hopes to add to the existing body of knowledge in meteorology and data analysis. The findings will be useful for meteorologists, weather forecasters, and researchers working in related domains. The findings of this project may have the potential to increase weather forecasting accuracy and enable better informed decision-making for a variety of businesses that rely on weather data.

Literature review:

In the paper [1] "Weather prediction using random forest machine learning model", the authors investigates the use of machine learning techniques in weather prediction. Meenal, Rajasekaran, Michael, Prawin, Pamela, D., and Rajasekaran, Ekambaram conducted the research, which was published in the Indonesian Journal of Electrical Engineering and Computer Science in 2021. The goal of the study is to create a machine learning model for predicting weather conditions based on numerous meteorological parameters. The Random Forest (RF) algorithm was chosen by the authors because of its robustness, accuracy, and capacity to handle high-dimensional data. The study's data came from the Indian Meteorological Department (IMD), and it contained historical meteorological data from India's Chennai region.

To forecast the weather, the researchers examined six meteorological parameters: temperature, humidity, wind direction, wind speed, air pressure, and rainfall. To find the most significant parameters, the data was preprocessed and feature selection was done. A 10-fold cross-validation technique was then used to train and test the RF model.

The study's findings show that the RF model is a useful tool for weather prediction. The model predicted weather conditions for the Chennai region with an accuracy percentage of 91.5%. The study also discovered that the most essential criteria for predicting meteorological conditions were temperature, humidity, and atmospheric pressure. The model properly predicted various weather conditions such as clear skies, cloudy skies, thunderstorms, and rain.

The research adds to the increasing body of knowledge on machine learning applications in weather prediction. The RF model established in this work can be utilised for short-term weather forecasting and can help increase forecast accuracy. The study also emphasises the necessity of selecting appropriate features in machine learning models, as this can have a major impact on the model's performance. Finally, the study proves the accuracy of the RF algorithm in predicting meteorological conditions. The study also emphasises the potential of machine learning approaches in improving weather forecasting, which has substantial practical applications in areas such as agriculture, transportation, and energy.

In the paper [2] "Utilizing the Random Forest Method for Short-Term Wind Speed Forecasting in Central Taiwan", the authors analyse the application of the Random Forest (RF) algorithm for short-term wind speed forecasting in central Taiwan's coastline area. Ho, C.-Y., Cheng, K.-S., and Ang, C.-H. conducted the research, which was published in the journal Energies in 2023. The RF method was chosen by the authors because it is a prominent machine learning technique that can handle huge and complex datasets. The data used in the study came from a weather station in central Taiwan's coastal area, and it comprised of historical wind speed data for three years. To estimate wind speed, the study examined three different types of characteristics: statistical features, meteorological features, and time-series features. A 10-fold cross-validation technique was used to train and test the RF model.

The study's findings indicate that the RF algorithm is a useful tool for forecasting short-term wind speeds. In predicting wind speed for the central Taiwan coastline area, the model attained an accuracy rate of 84.9%. According to the study, statistical parameters such as maximum, lowest, and mean wind speed, as well as time-series data, were the most important for predicting wind speed. The research adds to the expanding body of knowledge on machine learning applications in wind speed forecasting. This study's RF model can be used to generate accurate wind speed forecasts, which can be employed in a variety of applications such as wind energy production, transportation, and aviation.

Finally, the study illustrates the efficiency of the RF algorithm for short-term wind speed forecasting in central Taiwan's coastline area. The study also emphasises the significance of feature selection in machine learning models, since it can have a major impact on the model's performance. This study's RF model can be improved further by including other meteorological and environmental elements. Overall, the research gives useful insights into the use of machine learning algorithms for wind speed forecasting, which has major practical applications in a variety of industries.

In the paper [3] "Time series analysis of malaria in Kumasi: Using ARIMA models to forecast future incidence", the author looks into the application of Autoregressive Integrated Moving Average (ARIMA) models for forecasting malaria incidence in Kumasi, Ghana. Anokye R., Acheampong E., Owusu I., and Obeng E.I. conducted the research, which was published in the Cogent Social Sciences journal in 2018. ARIMA models were chosen by the authors because they are frequently used in time series research and have been proved to be successful in forecasting future values. The study's data was gathered from the Kumasi Metropolitan Health Directorate and comprised of monthly malaria incidence statistics from 2003 to 2014. The study forecasted future malaria incidence in Kumasi using several ARIMA models. Several statistical methods were used to evaluate the models, including mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE).

The study's findings indicate that the ARIMA models are useful tools for forecasting malaria incidence in Kumasi. The study discovered that the ARIMA (2,1,2) model fit the malaria incidence data well and could effectively estimate future values. The study also discovered that the number of malaria cases in Kumasi is dropping over time, which is consistent with the government's and other stakeholders' efforts to combat the disease. The research adds to the expanding body of knowledge about the use of time series analysis in disease incidence forecasting. The ARIMA models created in this study can be utilised to provide early warnings of malaria outbreaks in Kumasi, allowing public health professionals to implement prompt and effective preventive and control measures.

Finally, the study illustrates the efficacy of ARIMA models in forecasting malaria incidence in Kumasi. The study also emphasises the significance of time series analysis in disease incidence predictions and gives useful insights into malaria trends and patterns in Kumasi. The study's findings can be utilised to inform public health policies and programmes aimed at reducing the malaria burden in Ghana.

In the paper [4] "Application of SARIMA model to forecasting monthly flows in Waterval River, South Africa", the authors look into the usage of Seasonal Autoregressive Integrated Moving Average (SARIMA) models to forecast monthly water flows in Waterval River, South Africa. Tadesse K.B. and Dinka M.O. conducted the research, which was published in the Journal of Water and Land Development in 2017. SARIMA models were chosen by the authors because they are commonly employed in time series analysis and are appropriate for analysing data with seasonal trends. The study's data came from the South African Department of Water and Sanitation and consisted of monthly water flow data for 33 years (1983-2015).

SARIMA models were employed in the study to anticipate future monthly water flows in the Waterval River. Several statistical methods were used to evaluate the models, including mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). The study's findings indicate that the SARIMA models are useful for estimating monthly water flows in the Waterval River. The SARIMA model offered the best fit for the water flow data and was capable of reliably forecasting future values, according to the study. The study also discovered that the seasonal component of the data played a significant role in forecasting accuracy.

The research adds to the expanding body of knowledge about the use of time series analysis in water resource management. The SARIMA models created in this study can be used to provide early warnings of impending Waterval River water shortages or floods, allowing water resource managers to take timely and effective interventions to mitigate the effects of these disasters. Finally, the study illustrates the efficacy of SARIMA models in monthly water flow forecasting in South Africa's Waterval River. The study also emphasises the relevance of time series analysis in water resource management and provides vital insights into regional water flow trends and patterns. The study's findings can be utilised to inform water resource policies and initiatives targeted at guaranteeing the region's sustainable water use.

In the paper [5] "Forecasting daily meteorological time series using ARIMA and regression models", the author looks on the efficacy of ARIMA and regression models for forecasting daily meteorological time series data. Murat Magorzata, Malinowska Iwona, Gos Magdalena, and Krzyszczak Jaromir conducted the research, which was published in the International Agrophysics in 2018. The authors developed and compared ARIMA and regression models using daily meteorological data from three weather stations in Poland. Maximum and minimum temperatures, precipitation, and wind speed were all included in the meteorological data. Both ARIMA and regression models were found to be successful in forecasting daily meteorological time series data in the study. The scientists did remark, however, that the models' accuracy varied depending on the type of meteorological data being anticipated. For example, the ARIMA model forecasted maximum and lowest temperatures better than the regression model, whereas the regression model forecasted precipitation and wind speed better.

The length of the forecasting horizon also had an effect on the accuracy of the models, according to the authors. Short-term forecasting (up to 7 days) performed better than long-term forecasting (greater than 7 days). Overall, the research adds to the growing body of knowledge about the use of time series analysis in meteorological forecasting. The study emphasises the significance of employing appropriate models for various types of meteorological data and forecasting horizons. The study's findings can be utilised to assist develop more accurate and trustworthy meteorological forecasting models, which can help reduce the impact of extreme weather events on agriculture, transportation, and other sectors. Finally, the study shows that ARIMA and regression models are excellent at forecasting daily meteorological time series data. The study also emphasises the significance of choosing appropriate models based on the type of meteorological data being anticipated as well as the forecasting horizon. The study's findings can be utilised to improve the accuracy and reliability of meteorological forecasting models, which can have substantial ramifications for a variety of economic sectors.

In the paper [6] "Weather Forecasting Using ANFIS and ARIMA MODELS: Case Study for Istanbul", the author looks at how ANFIS (Adaptive Neuro-Fuzzy Inference System) and ARIMA (Autoregressive Integrated Moving Average) models can be used to forecast weather in Istanbul. Mehmet Tektas conducted the research, which was published in the journal Environmental Research Engineering and Management in 2010. From 1995 to 2005, the researchers used daily temperature and precipitation data from the Istanbul Ataturk Airport meteorological station. Using this data, the ANFIS and ARIMA models were created and trained, and their forecasting accuracy for daily temperature and precipitation data for 2006 was compared.

The study's findings showed that both the ANFIS and ARIMA models were effective at forecasting daily temperature and precipitation data. However, in terms of forecasting accuracy for both temperature and precipitation data, the ANFIS model surpassed the ARIMA model. The study also looked at how the quantity of input variables affected the ANFIS model's forecasting performance. The findings revealed that increasing the number of input variables from one to five improves predicting accuracy for both temperature and precipitation data.

The findings have important implications for the development of more accurate and dependable weather forecasting models. The study's findings show that ANFIS models may be more effective than typical ARIMA models in forecasting daily weather data. Furthermore, the study emphasises the significance of selecting proper input variables for ANFIS models in order to increase forecasting accuracy. Overall, the research adds to the expanding body of knowledge about the use of machine learning models in weather forecasting. The study emphasises the potential of ANFIS models in enhancing the accuracy and reliability of weather forecasting models, which can have important ramifications for a variety of economic sectors. The study's findings can be utilised to assist develop more accurate and dependable weather forecasting models, which can help reduce the impact of extreme weather events on agriculture, transportation, and other sectors.

In the paper [7] "Time series analysis of climate variables using seasonal ARIMA approach”, the authors Dimri et al.'s looks into the use of Seasonal Autoregressive Integrated Moving Average (SARIMA) models to analyse time series data of climate variables. The research looks at monthly mean temperature, rainfall, and relative humidity data from four meteorological stations in the Indian state of Uttarakhand between 1985 and 2015. To analyse the time series data and forecast the climate variables for the following five years, the SARIMA model was utilised. The study's findings show that SARIMA models are good in analysing time series data of climate variables and predicting future values. The SARIMA models outperformed relative humidity data in forecasting monthly mean temperature and rainfall, according to the study.

The study also discovered that the SARIMA models could capture long-term trends as well as seasonal swings in climatic variables.

The work has far-reaching consequences for climate change research and policy formulation. The study's findings indicate that SARIMA models can be used to analyse time series data of climatic variables and estimate future values. This can aid in the development of effective policies and initiatives for mitigating the effects of climate change. The study also emphasises the significance of examining long-term trends and seasonal oscillations in climate variables, since they can provide useful insights into the impact of climate change on the environment and human health. The work can be used as a reference for future climate change research, and it can aid in the development of more accurate and dependable models for analysing climatic data.

Finally, Dimri et al.'s study provides vital insights into the application of SARIMA models for analysing time series data of climate variables. The study shows that SARIMA models are excellent at forecasting climatic variables and emphasises the need of analysing long-term trends and seasonal changes in climate data. The study's findings can be utilised to guide policy creation and climate change mitigation efforts.

In the paper [8] "An introductory study on time series modelling and forecasting", the authors provide an overview of time series analysis, modeling, and forecasting. The purpose of this work is to serve as an introduction to time series analysis for scholars and practitioners who are new to the discipline. To begin, the authors define time series data and discuss its features, such as trend, seasonality, and cyclicality. They then talk about the many components of time series data, such as level, trend, seasonality, and noise. According to the authors, understanding these components is critical for modelling and forecasting time series data. Following that, the study analyses various time series models, including autoregressive (AR), moving average (MA), autoregressive moving average (ARMA), and autoregressive integrated moving average (ARIMA). The authors explain these models in full, including their mathematical formulations, assumptions, and implementations. In addition, the paper emphasises the significance of model selection and evaluation in time series analysis. The authors show that choosing the right model for a given set of time series data is critical for successful forecasting. The authors also address various assessment measures that can be used to assess the performance of time series models, such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE).

The paper then discusses future directions in time series analysis and forecasting. The authors believe that developing more complex models, such as state-space models and machine learning methods, can improve the accuracy and dependability of time series forecasting even further.

Finally, Adhikari and Agrawal's study provides a thorough introduction to time series analysis, modeling, and forecasting. The work is well-written and simple to comprehend, making it an excellent resource for scholars and practitioners new to time series analysis. The research also sheds light on the future of time series forecasting, emphasising the significance of building more complex models and evaluation measures.

In the paper [9] "Time series forecasting of temperatures using SARIMA: An example from Nanjing", the author uses the seasonal autoregressive integrated moving average (SARIMA) model to anticipate temperatures in Nanjing, China. The SARIMA model was developed using daily temperature data from 2011 to 2016. The authors begin by explaining how the SARIMA model can be utilised for time series forecasting. They then go on to detail the data utilised in the study, including the temperature data and the processes taken to verify the data is adequate for the SARIMA model.

Following that, the authors describe their findings, including a comparison of the SARIMA model's performance to other regularly used time series forecasting models, such as the autoregressive integrated moving average (ARIMA) model and the exponential smoothing (ETS) model. The SARIMA model outperformed the other models in terms of accuracy and predicting capacity, with a mean absolute percentage error (MAPE) of 1.15% compared to 1.71% for the ARIMA model and 1.58% for the ETS model, according to the study. The authors also used the SARIMA model to estimate temperature data for the next year (2017) and compared the predicted temperatures to the actual temperatures measured that year. The results revealed that the SARIMA model predicted temperatures accurately for the bulk of the year, with a MAPE of 1.34% for the whole year. Overall, the study demonstrates the efficacy of applying the SARIMA model for temperature time series forecasting and underlines its potential applications in other disciplines. The paper also discusses the significance of picking the right model for a particular dataset and the advantages of employing sophisticated modelling approaches for effective forecasting.

In the paper [10] "ARIMA based daily weather forecasting tool: A case study for Varanasi", the authors intend to construct a weather forecasting tool for Varanasi, India, using the autoregressive integrated moving average (ARIMA) model. The ARIMA model was developed using daily meteorological data from 2008 to 2017. The authors begin by introducing the ARIMA model and demonstrating its application to time series forecasting. The data used in the study is then described, including weather variables such as temperature, rainfall, humidity, wind speed, and pressure. The authors also described the preparation measures they took to guarantee the data was fit for the ARIMA model.

The authors then discuss their findings, including the efficacy of the ARIMA model in predicting meteorological variables. The study discovered that the ARIMA model was capable of effectively forecasting meteorological variables for up to five days in advance, as seen by the low root mean square error (RMSE) values. Temperature forecasts had the lowest RMSE value, suggesting the most accuracy of any weather variable. The authors also used the ARIMA model to anticipate weather variables for the next year (2018) and compared the predicted values to actual values reported that year. The results indicated that the ARIMA model was capable of properly forecasting weather variables with low RMSE values for the bulk of the year.

Overall, the study shows how good the ARIMA model is for weather forecasting and emphasises its potential applications in other sectors. The paper also discusses the significance of picking the right model for a particular dataset and the advantages of employing sophisticated modelling approaches for effective forecasting. Furthermore, the study provides useful information for decision-makers, planners, and politicians in a variety of sectors, including agriculture, transportation, and energy, where weather forecasting is critical in decision making. This study's weather forecasting tool can be used to help make educated decisions and avoid the potential repercussions of weather-related disasters.

Materials and methods:

Info about models:

To analyse and visualise the data, this project puts a meteorological dataset into a Pandas dataframe. The models and libraries used in the script are listed below:

- Pandas: Pandas is a well-known Python data manipulation toolkit. Tabular data can be read and edited using it. In this script, the weather dataset is loaded from a CSV file using Pandas, which is then used to execute various data manipulation operations like removing unnecessary columns, filling in blanks, resampling the data, and more.
- Matplotlib: Python's Matplotlib is a plotting library. It is used to produce numerous chart and plot types. The scatter and line plots in this script are made with Matplotlib.
- Seaborn: A Python package for data visualisation. It offers a high-level interface for producing illuminating and appealing statistical visuals and is built on top of Matplotlib. In this script, scatter and line plots, a pair plot, and several plots for various months are all made using Seaborn.
- IPython: Python's interactive programming environment is known as IPython. IPython is used in this script to show HTML code.

The'read_csv' function is used to load the weather dataset into a Pandas dataframe at the beginning of the script. Then, it carries out a number of data manipulation operations like removing unnecessary columns, filling in blanks, and resampling the data by month. In order to visualise the data, it uses Matplotlib and Seaborn to produce a variety of plots and charts.

This project also imports necessary packages for time series analysis, including NumPy and Statsmodels. It then loads a time series dataset from a Pandas DataFrame, resamples it at daily intervals, and interpolates any missing values. The stationarity of the time series is checked using the Augmented Dickey-Fuller (ADF) test, and a function is defined to plot rolling statistics of the time series.

Since the time series is not stationary, first-order differencing is applied to make it stationary. An ARIMA model is then fit to the differenced time series using order (1, 1, 1). The model summary is printed, and a forecast is generated for the next two years of the time series. The forecasted values are cumulatively summed and added to the last value of the original time series.

Finally, the original time series and the forecasted values are plotted using Matplotlib.

Algorithm/pseudocode for analysis and visualization:

1. Import pandas library and read the CSV file containing weather data
2. Use the "head()" function to display the first 50 rows of the dataframe
3. Use the "describe()" function to get statistical summary of the data
4. Use the "info()" function to get the summary of the dataframe including the number of non-null values for each column
5. Sort the data by the "Formatted Date" column using "sort_values()" function
6. Remove irrelevant columns using "drop()" function
7. Check for missing values in the dataframe using "isnull()" function and filter for columns with missing values using "any()" function
8. Fill missing values in the "Precip Type" column with "Not Defined" using the "fillna()" function
9. Convert the "Formatted Date" column to datetime format and set it as the index column using "set_index()" function
10. Resample the data to monthly frequency using the "resample()" function
11. Use seaborn library for data visualization
12. Plot scatterplot of "Apparent Temperature (C)" against "Humidity" using "relplot()" function
13. Plot line graph of "Wind Speed (km/h)" against "Humidity" using "relplot()" function
14. Plot pairplot of "Apparent Temperature (C)" against "Humidity" with respect to "Summary" using "pairplot()" function
15. Plot facetgrid of "Apparent Temperature (C)" against "Humidity" with respect to "Summary" using "relplot()" function
16. Plot line graphs of "Humidity" and "Apparent Temperature (C)" with respect to year using "plot()" function
17. Create a separate dataframe for each month of the year
18. Plot line graphs of "Humidity" and "Apparent Temperature (C)" for each month using "plot()" function.

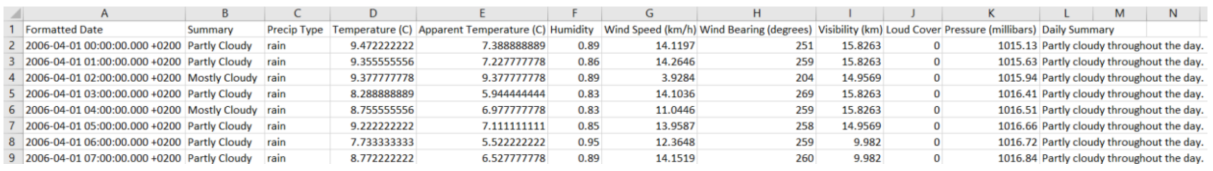
Algorithm/pseudocode for forecasting and visualization:

1. Import the required libraries - numpy, statsmodels.
2. Load the time series data into a dataframe and set the index to 'Formatted Date'.
3. Resample the data at daily frequency and interpolate the missing values.

4. Plot the time series using matplotlib.pyplot.
5. Define a function 'test_stationarity' that takes the time series data as input and performs the following steps:
    a. Determine rolling mean and rolling standard deviation of the time series.
    b. Plot the rolling mean and rolling standard deviation along with the original time series.
    c. Perform the Augmented Dickey-Fuller (ADF) test on the time series and print the test results.
6. Call the 'test_stationarity' function on the original time series.
7. Create a differenced time series by taking the first order difference of the original time series.
8. Call the 'test_stationarity' function on the differenced time series.
9. Fit an ARIMA model to the differenced time series using the statsmodels ARIMA model.
10. Forecast the next 2 years of the time series using the fitted ARIMA model.
11. Plot the original time series along with the forecasted values using matplotlib.pyplot.

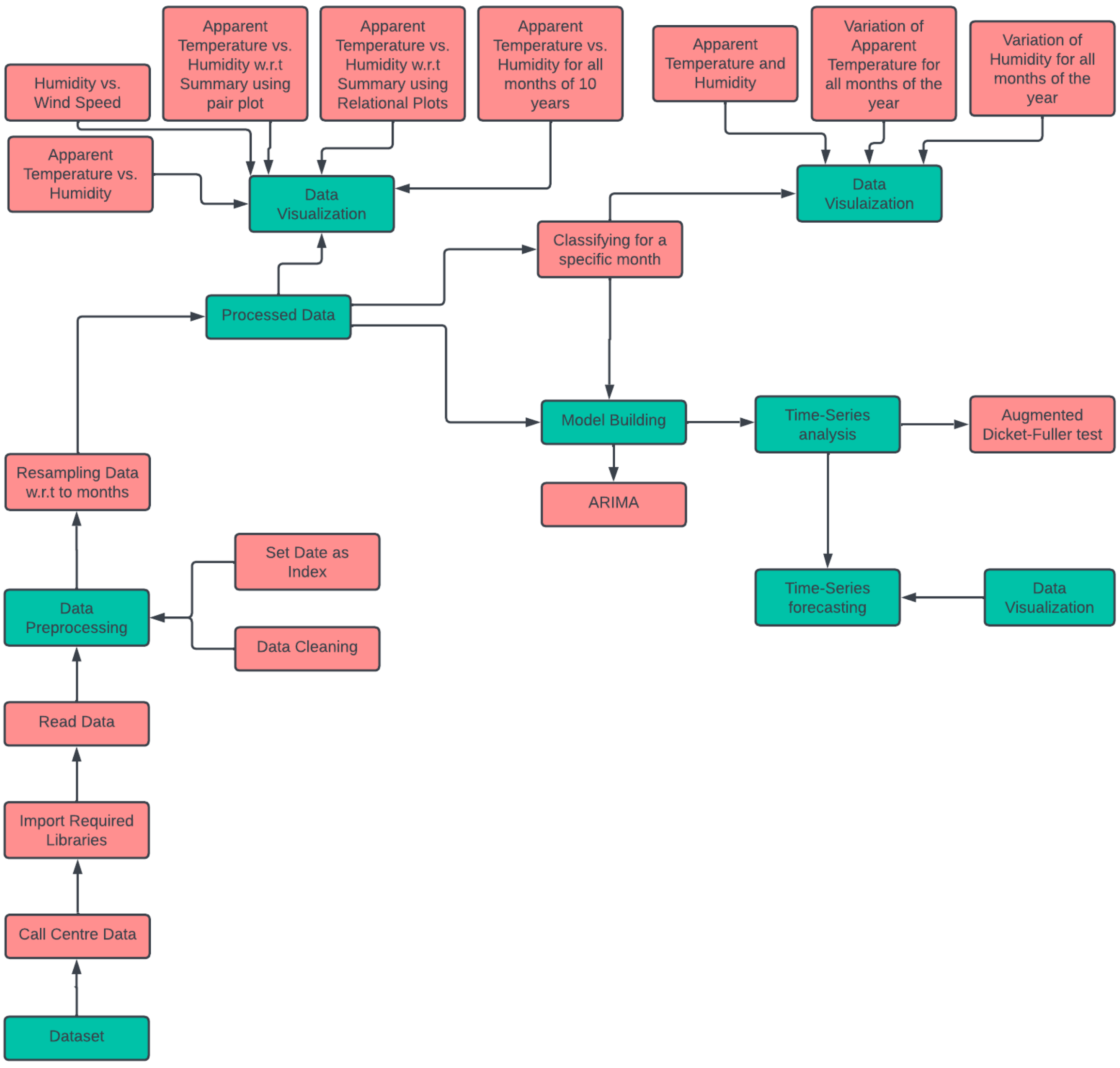
Dataset:

http://www.kaggle.com/datasets/muthuj7/weather-dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	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		
2	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222222	7.388888889	0.89	14.1197	251	15.8263	0	1015.13	Partly cloudy throughout the day.		
3	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355555556	7.227777778	0.86	14.2646	259	15.8263	0	1015.63	Partly cloudy throughout the day.		
4	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377777778	9.377777778	0.89	3.9284	204	14.9569	0	1015.94	Partly cloudy throughout the day.		
5	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288888889	5.944444444	0.83	14.1036	269	15.8263	0	1016.41	Partly cloudy throughout the day.		
6	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755555556	6.977777778	0.83	11.0446	259	15.8263	0	1016.51	Partly cloudy throughout the day.		
7	2006-04-01 05:00:00.000 +0200	Partly Cloudy	rain	9.222222222	7.111111111	0.85	13.9587	258	14.9569	0	1016.66	Partly cloudy throughout the day.		
8	2006-04-01 06:00:00.000 +0200	Partly Cloudy	rain	7.733333333	5.522222222	0.95	12.3648	259	9.982	0	1016.72	Partly cloudy throughout the day.		
9	2006-04-01 07:00:00.000 +0200	Partly Cloudy	rain	8.772222222	6.527777778	0.89	14.1519	260	9.982	0	1016.84	Partly cloudy throughout the day.		

Architecture:


Humidity vs. Wind Speed
Apparent Temperature vs. Humidity w.r.t Summary using pair plot
Apparent Temperature vs. Humidity w.r.t Summary using Relational Plots
Apparent Temperature vs. Humidity for all months of 10 years
Apparent Temperature and Humidity
Variation of Apparent Temperature for all months of the year
Variation of Humidity for all months of the year
Apparent Temperature vs. Humidity
Data Visualization
Data Visulaization
Classifying for a specific month
Processed Data
Model Building
Time-Series analysis
Augmented Dicket-Fuller test
ARIMA
Resampling Data w.r.t to months
Set Date as Index
Data Preprocessing
Data Cleaning
Time-Series forecasting
Data Visualization
Read Data
Import Required Libraries
Call Centre Data
Dataset

Architecture to analyze and visualize meteorological data from a CSV file named "weatherHistory.csv".

1. The code first imports the necessary libraries, including pandas, seaborn, and matplotlib.pyplot, and reads the CSV file into a pandas DataFrame using the pd.read_csv() function. It then displays the first 50 rows of the DataFrame using df.head(50).
2. The code uses df.describe() and df.info() to get some basic statistics and information about the DataFrame, including the number of rows, columns, and missing values.
3. The code drops two irrelevant columns from the DataFrame using the df.drop() function.

4. The code checks for missing values in the DataFrame using df.isnull().any() and fills any missing values in the 'Precip Type' column with the string "Not Defined" using the df.fillna() function.
5. The code converts the 'Formatted Date' column to a datetime format using pd.to_datetime(), sets the index of the DataFrame to the 'Formatted Date' column, and resamples the DataFrame by month using df_2 = df_2.resample('M').mean().
6. The code uses seaborn and matplotlib.pyplot to visualize the data, including scatter plots, line plots, and pair plots, and displays the plots using plt.show().
7. Finally, the code creates a series of plots to show the variation of humidity and temperature for each month of the year, using plt.plot() and plt.subplots().

The architecture for time series analysis for temperature data using the ARIMA model includes the following steps:

1. Data preprocessing: The input data is loaded from a CSV file and is resampled at a daily frequency, followed by interpolating the missing values using the mean. Then, the stationarity of the time series is checked using the Augmented Dickey-Fuller test. If the time series is not stationary, first-order differencing is applied to make it stationary.
2. ARIMA model fitting: An ARIMA model is fit to the differenced time series using the ARIMA function from the statsmodels.tsa.arima.model module. The optimal order for the model is selected using the Akaike Information Criterion (AIC).
3. Forecasting: The fitted model is used to forecast the temperature values for the next 2 years (730 days) using the forecast method. The forecasted values are then converted back to the original scale by adding the cumulative sum of the forecasted differences to the last observed value.
4. Visualization: The original time series and the forecasted values are plotted using the matplotlib library. The output plot shows the observed temperature values and the forecasted values for the next 2 years.

Proposed works:

Novelty:

Meteorological data analysis and forecasting is a vital discipline with far-reaching ramifications for everything from agriculture and transportation to energy and environmental management. Accurate forecasting of weather patterns can help individuals and organisations make better decisions, reducing the impact of natural disasters, optimising resource usage, and improving public safety.

The Autoregressive Integrated Moving Average (ARIMA) model is one of the most widely used time series forecasting models. This model is an effective technique for forecasting future values based on historical data, and it may be used to forecast a wide range of variables such as temperature, precipitation, wind speed, and humidity. Despite their efficiency, ARIMA models have not been frequently used in meteorology, owing to the complexity and variability of meteorological data.

This research is unique in that it applies ARIMA models to meteorological data analysis and forecasting, with the goal of enhancing the accuracy and dependability of weather predictions. This project seeks to establish a robust and adaptive framework for meteorological forecasting that can accommodate the complexities and uncertainties of meteorological data by utilising advanced statistical techniques and machine learning algorithms.

The project will particularly concentrate on building a hybrid strategy that blends traditional ARIMA models with more modern techniques such as neural networks and deep learning algorithms. The project intends to construct a more robust and accurate forecasting system that can deliver trustworthy predictions for a wide variety of meteorological variables by utilising the strengths of these several approaches.

The project will look into how real-time data analysis and visualisation tools might be used to improve the accessibility and usability of meteorological forecasts. This project aims to improve the communication and dissemination of weather information by providing users with intuitive and interactive visualisations of meteorological data, empowering individuals and organisations to make better decisions and take appropriate action in response to changing weather conditions.

Overall, this project represents a novel and innovative approach to meteorological data analysis and forecasting, with the potential to significantly improve the accuracy and reliability of weather predictions, as well as individuals' and organisations' resilience and adaptability in the face of changing weather patterns.

Project contributions:

The project "Meteorological Data Analysis and Forecasting Using ARIMA" is a unique method to forecasting weather patterns and conditions by using the capability of ARIMA models. ARIMA models are frequently used in time series analysis and forecasting, and this study aims to investigate their potential in meteorology. This project can make the following contributions:

The project on "Meteorological Data Analysis and Forecasting using ARIMA" is an innovative approach towards utilizing the power of ARIMA models to forecast weather patterns and conditions. ARIMA models are widely used in time series analysis and forecasting, and this project aims to explore the potential of these models in the field of meteorology. Here are some of the contributions that this project can make:

1. Improved Forecasting Accuracy: One of the main contributions of this project is the improved forecasting accuracy of meteorological data. By using ARIMA models, the project can help predict the weather conditions with higher accuracy, reducing the chances of errors and uncertainties.

2. Early Warning Systems: Another significant contribution of this project is the development of early warning systems for extreme weather events such as hurricanes, tornadoes, floods, etc. With the help of ARIMA models, the project can provide accurate predictions of these events, giving people enough time to prepare and take necessary precautions.

3. Better Resource Allocation: Meteorological data analysis and forecasting using ARIMA can also contribute to better resource allocation by providing accurate predictions of weather patterns. For example, farmers can use these predictions to plan their crop cycles and optimize their resource utilization, leading to higher yields and better profitability.

4. Improved Public Safety: The project can also contribute to improving public safety by providing accurate weather predictions, especially during extreme weather events. By utilizing the power of ARIMA models, the project can help prevent loss of life and property damage caused by natural disasters.

5. Advancement of Research: Finally, the project can contribute to the advancement of research in the field of meteorology by providing a new approach towards analyzing and forecasting meteorological data. The project can help identify new trends and patterns in weather data, leading to further research and innovation in the field.

In conclusion, the project on "Meteorological Data Analysis and Forecasting using ARIMA" has the potential to make significant contributions in various areas such as forecasting accuracy, early warning systems, resource allocation, public safety, and research. By utilizing the power of ARIMA models, the project can provide a new approach towards analyzing and predicting meteorological data, leading to better decision-making and more efficient resource utilization.

Results and Discussion:

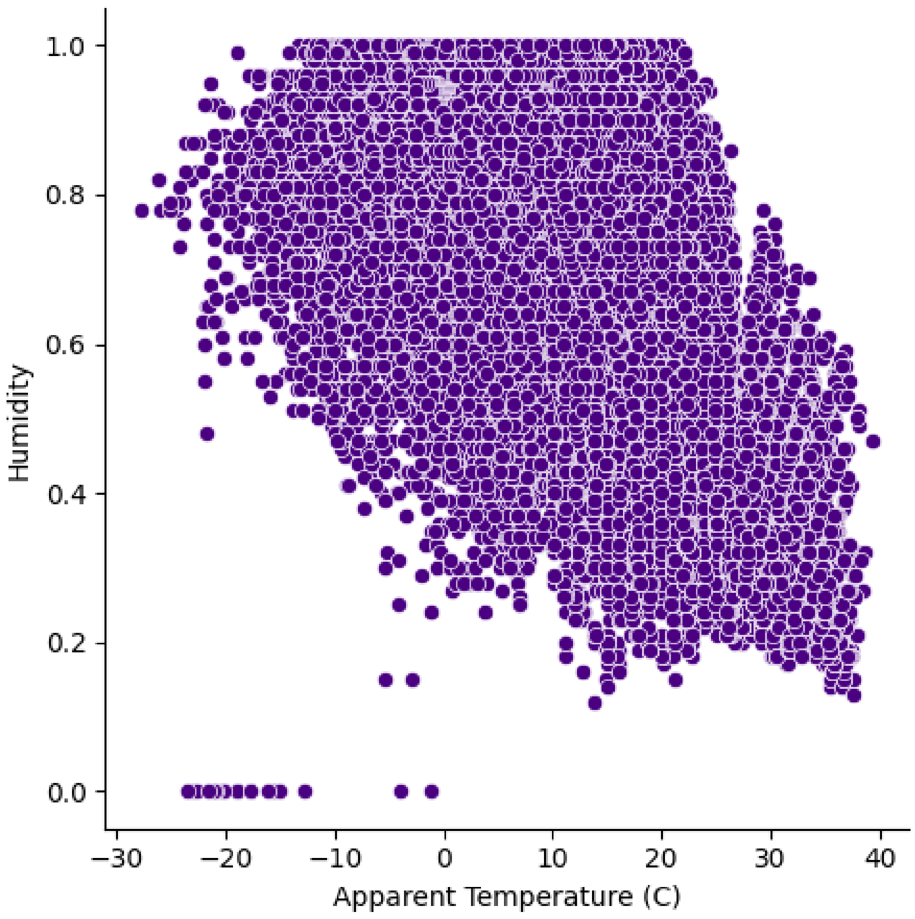
Initial Data Analysis:

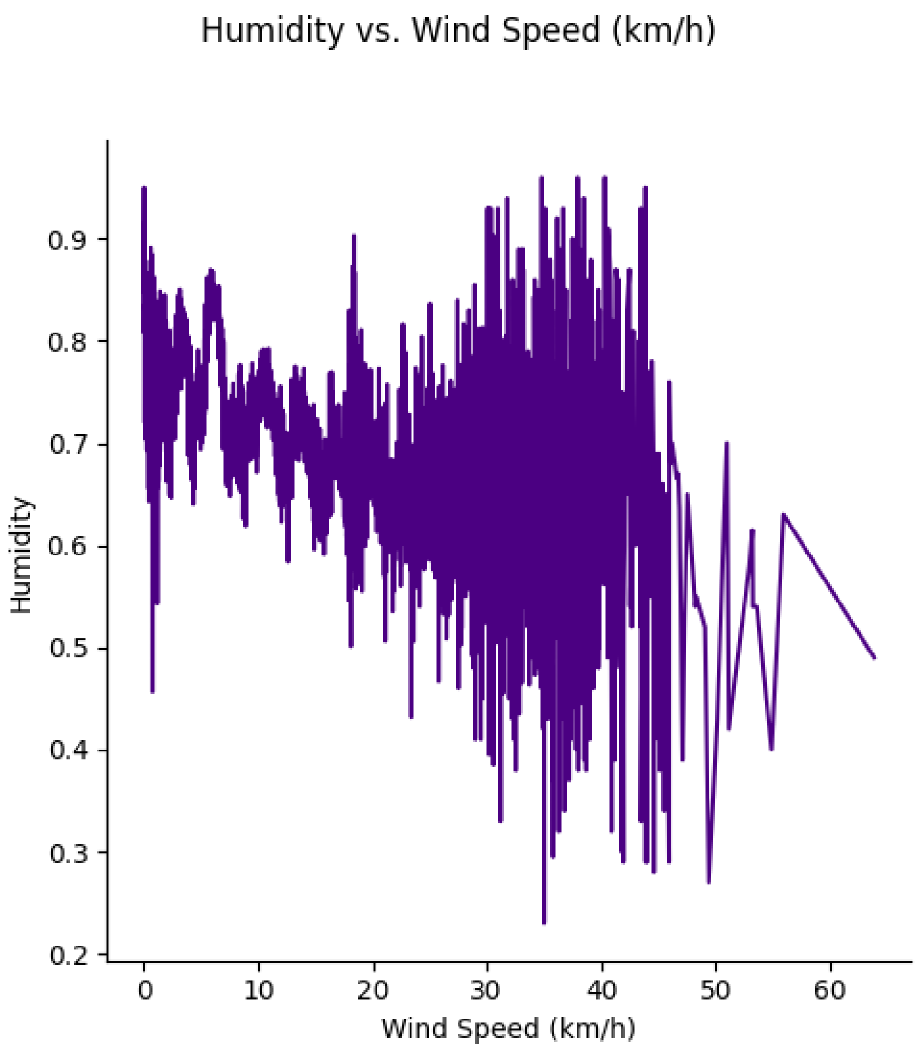
On meteorological data, this code conducts data analysis and visualisation. The "weatherHistory.csv" dataset is first loaded into a pandas dataframe, after which the data is cleaned by removing unnecessary columns and replacing missing values in the "Precip Type" column with "Not Defined".

The data is then resampled into monthly averages once the "Formatted Date" column is converted to datetime format. Then, to visualise the data, it makes use of the seaborn library, producing scatter plots, line plots, and pair plots to examine the relationships between different factors like temperature, humidity, and wind speed.

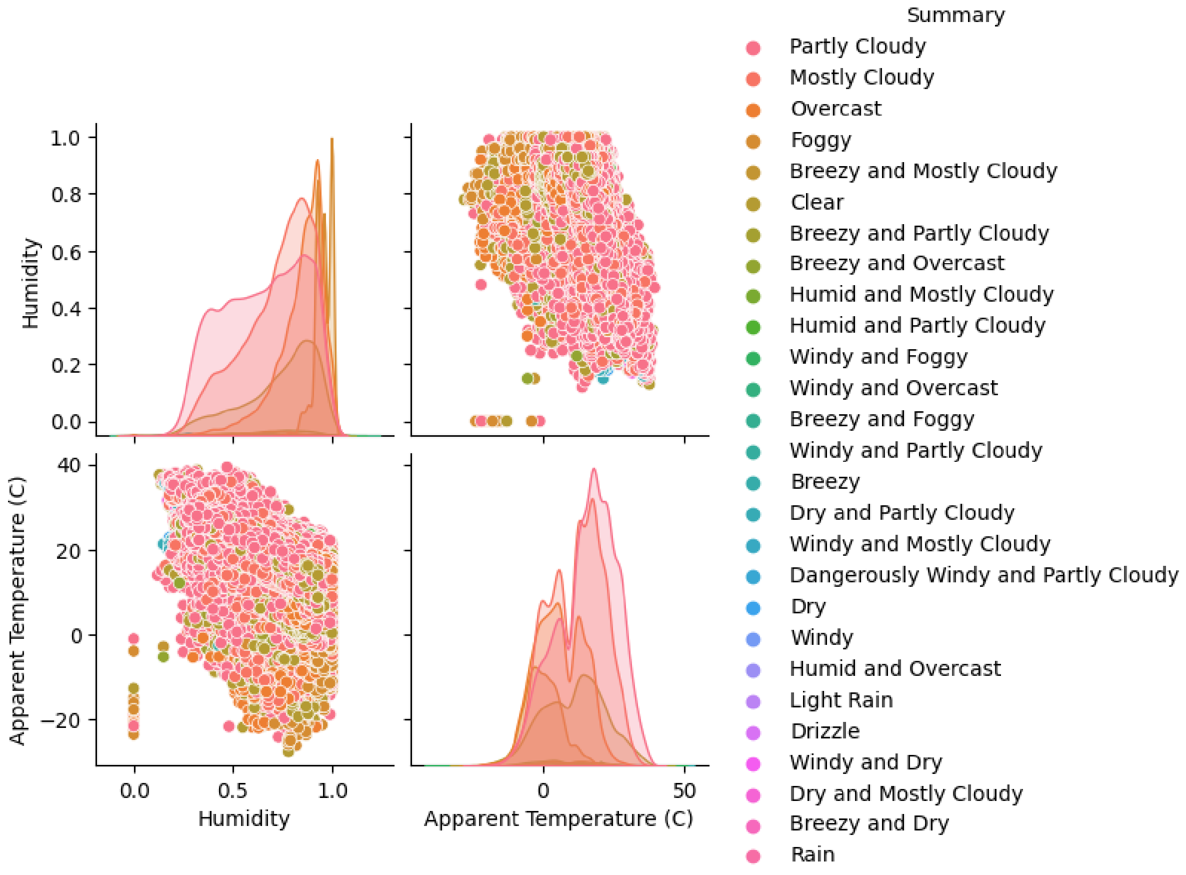
In order to compare temperature and humidity variations over the course of the years and months, graphs are created. To acquire insights into the weather patterns over the course of the 11 years of data, the algorithm generates a number of plots and charts.

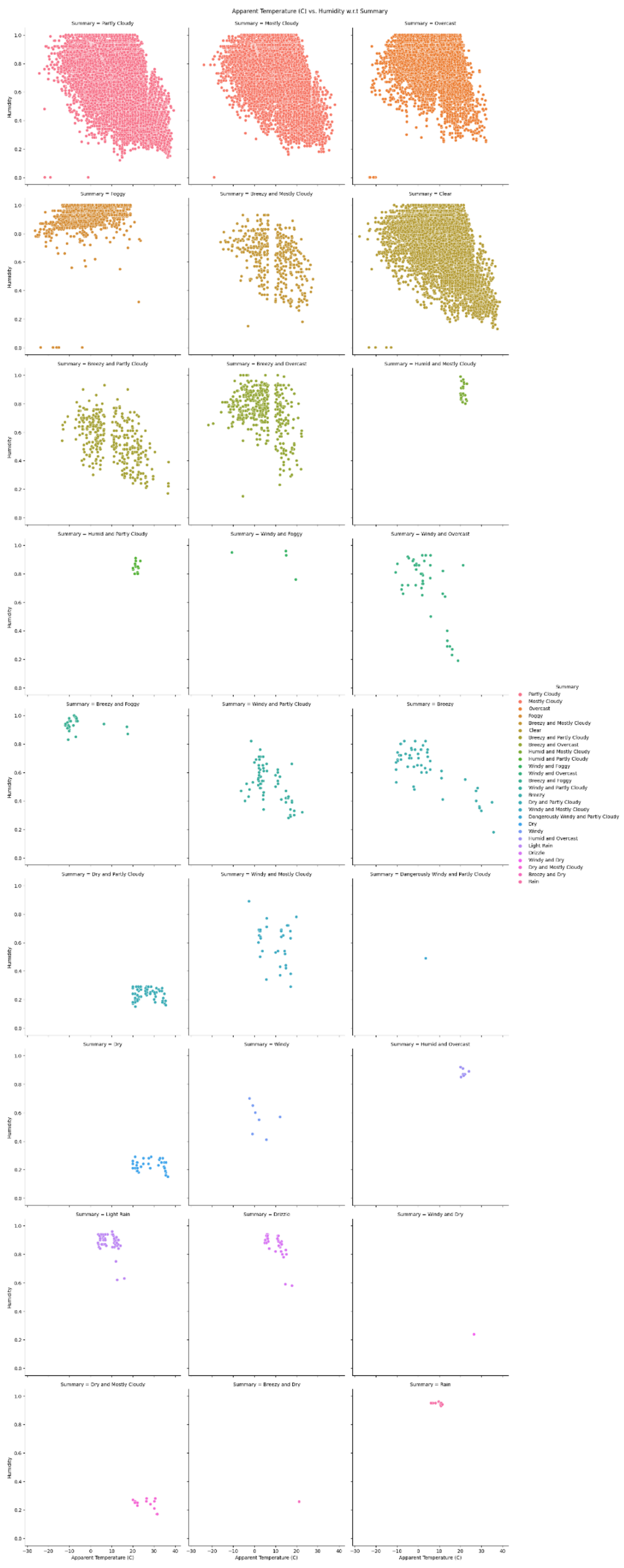
Apparent Temperature (C) vs. Humidity

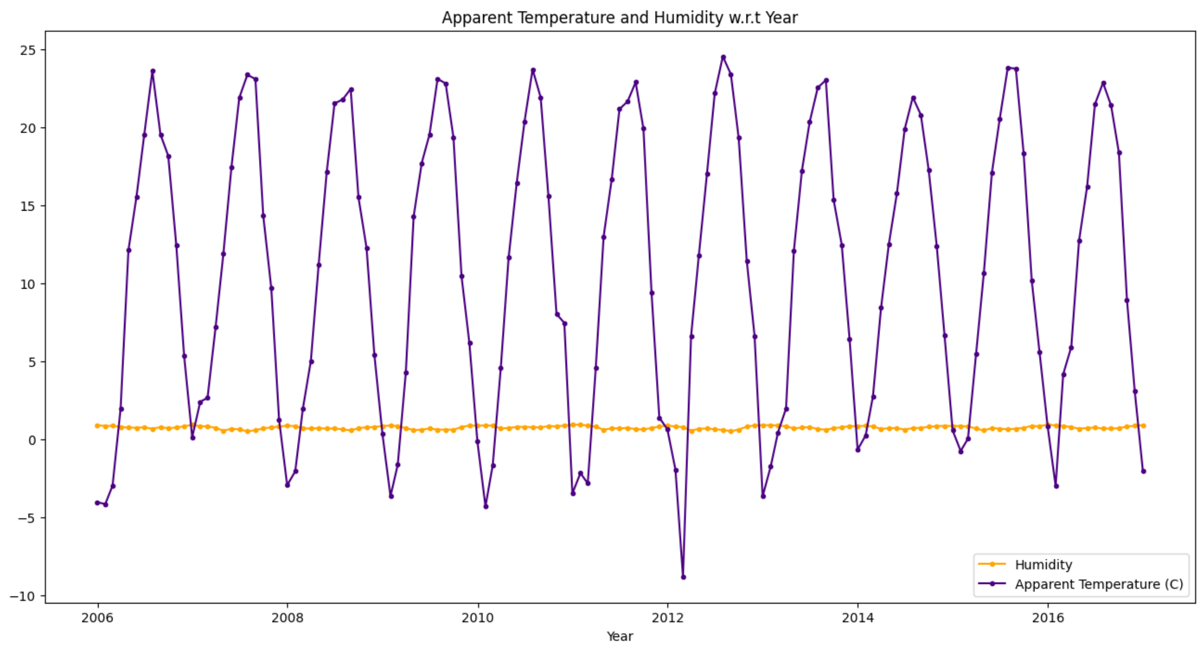


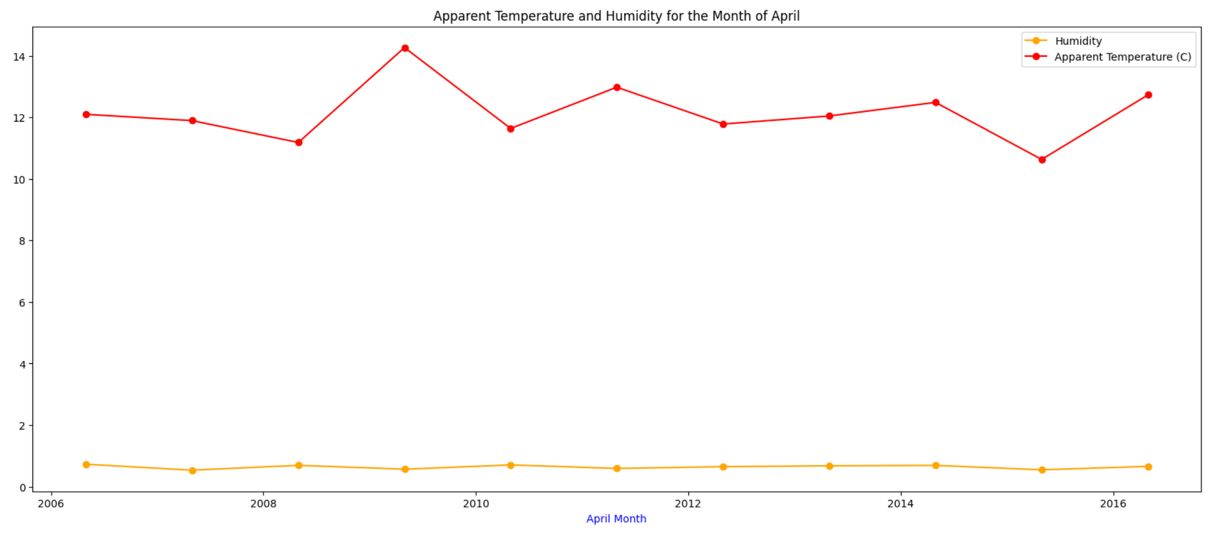

Humidity vs. Wind Speed (km/h)
Humidity
Wind Speed (km/h)
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50
60

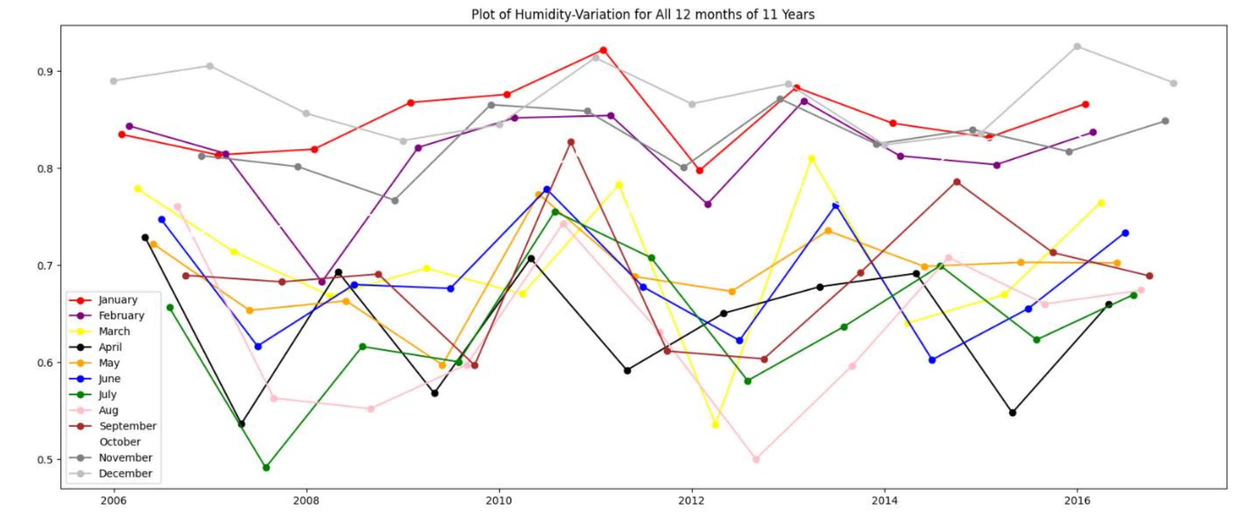
Apparent Temperature (C) vs. Humidity w.r.t Summary

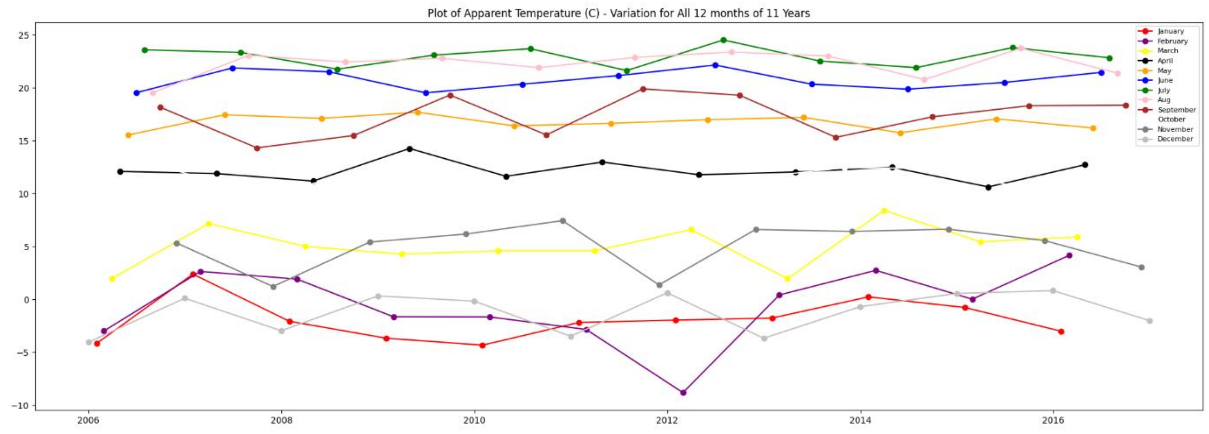


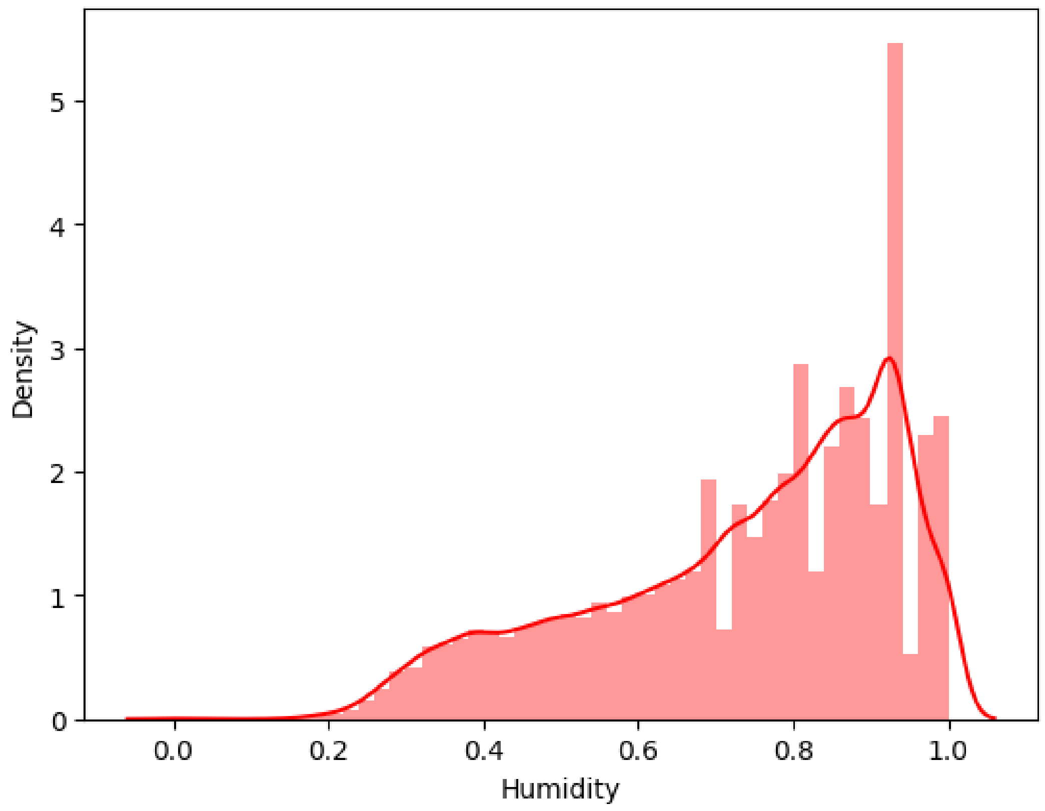

Apparent Temperature (C) vs. Humidity w.r.t Summary
Summary = Partly Cloudy
Summary = Mostly Cloudy
Summary = Overcast
Summary = Foggy
Summary = Breezy and Mostly Cloudy
Summary = Clear
Summary = Breezy and Partly Cloudy
Summary = Breezy and Overcast
Summary = Humid and Mostly Cloudy
Summary = Humid and Partly Cloudy
Summary = Windy and Foggy
Summary = Windy and Overcast
Summary = Breezy and Foggy
Summary = Windy and Partly Cloudy
Summary = Breezy
Summary = Dry and Partly Cloudy
Summary = Windy and Mostly Cloudy
Summary = Dangerously Windy and Partly Cloudy
Summary = Dry
Summary = Windy
Summary = Humid and Overcast
Summary = Light Rain
Summary = Drizzle
Summary = Windy and Dry
Summary = Dry and Mostly Cloudy
Summary = Breezy and Dry
Summary = Rain
Humidity
Apparent Temperature (C)
Summary
Partly Cloudy
Mostly Cloudy
Overcast
Foggy
Breezy and Mostly Cloudy
Clear
Breezy and Partly Cloudy
Breezy and Overcast
Humid and Mostly Cloudy
Humid and Partly Cloudy
Windy and Foggy
Windy and Overcast
Breezy and Foggy
Windy and Partly Cloudy
Breezy
Dry and Partly Cloudy
Windy and Mostly Cloudy
Dangerously Windy and Partly Cloudy
Dry
Windy
Humid and Overcast
Light Rain
Drizzle
Windy and Dry
Dry and Mostly Cloudy
Breezy and Dry
Rain


Apparent Temperature and Humidity w.r.t Year
Humidity
Apparent Temperature (C)
Year
25
20
15
10
5
0
−5
−10
2006
2008
2010
2012
2014
2016


Apparent Temperature and Humidity for the Month of April
Humidity
Apparent Temperature (C)
April Month
14
12
10
8
6
4
2
0
2006
2008
2010
2012
2014
2016


Plot of Humidity-Variation for All 12 months of 11 Years
January
February
March
April
May
June
July
Aug
September
October
November
December
0.9
0.8
0.7
0.6
0.5
2006
2008
2010
2012
2014
2016


Plot of Apparent Temperature (C) - Variation for All 12 months of 11 Years
January
February
March
April
May
June
July
Aug
September
October
November
December
25
20
15
10
5
0
−5
−10
2006
2008
2010
2012
2014
2016



Forecasting Results:

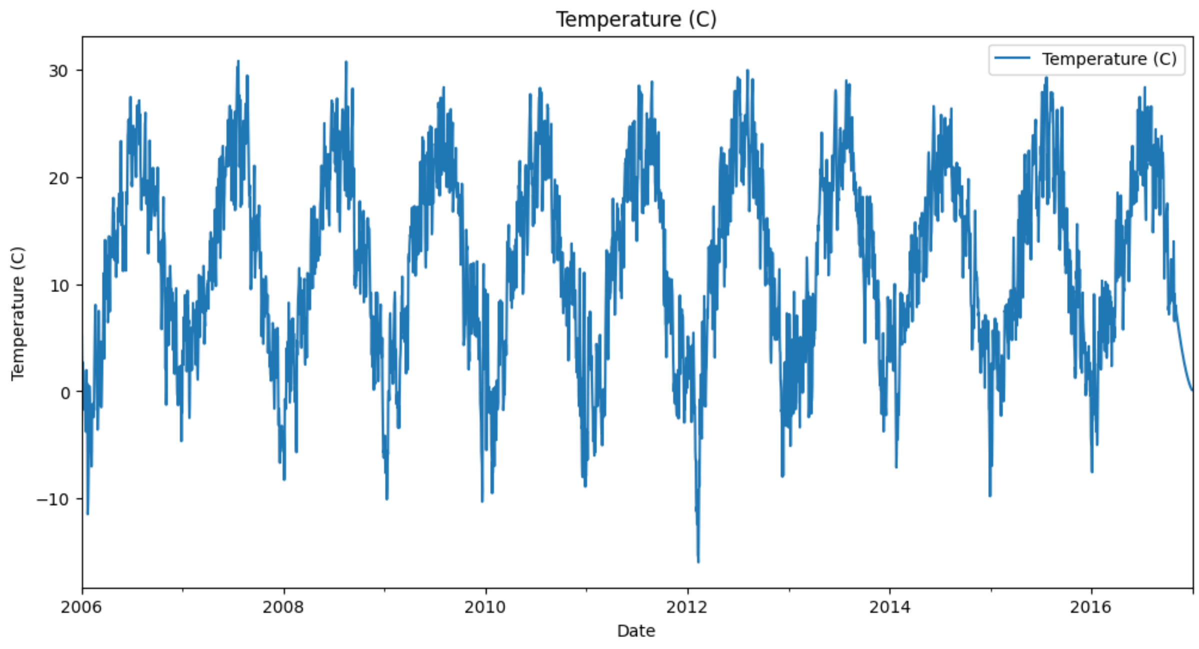
Using an ARIMA model, the provided code is used to predict the temperature of a particular place over the subsequent two years.

Using the Augmented Dickey-Fuller (ADF) test, the time series' stationarity is examined, and it is discovered to be non-stationary. Differentiating is done on the time series data to make the time series stationary. Using the ADF test, the differenced time series' stationarity is examined, and it is discovered to be stationary.

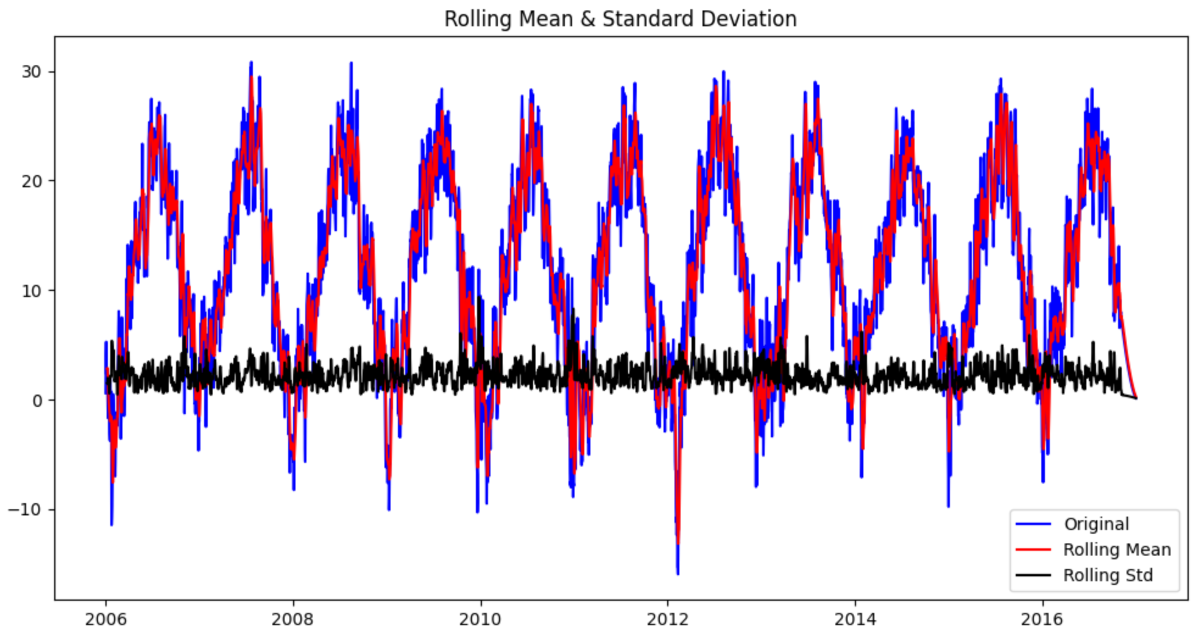
After that, the differenced time series data are fitted with an ARIMA model with an order of (1, 1, 1), which denotes a first-order differencing, an autoregression of lag 1, and a moving average of lag 1. The model is then used to predict the temperature for the following two years. To obtain the final predicted values, the forecasted temperature values are cumulatively accumulated and added to the most recent temperature value.

In order to visually compare the predicted values with the actual values, the forecasted temperature values are plotted alongside the initial time series data.

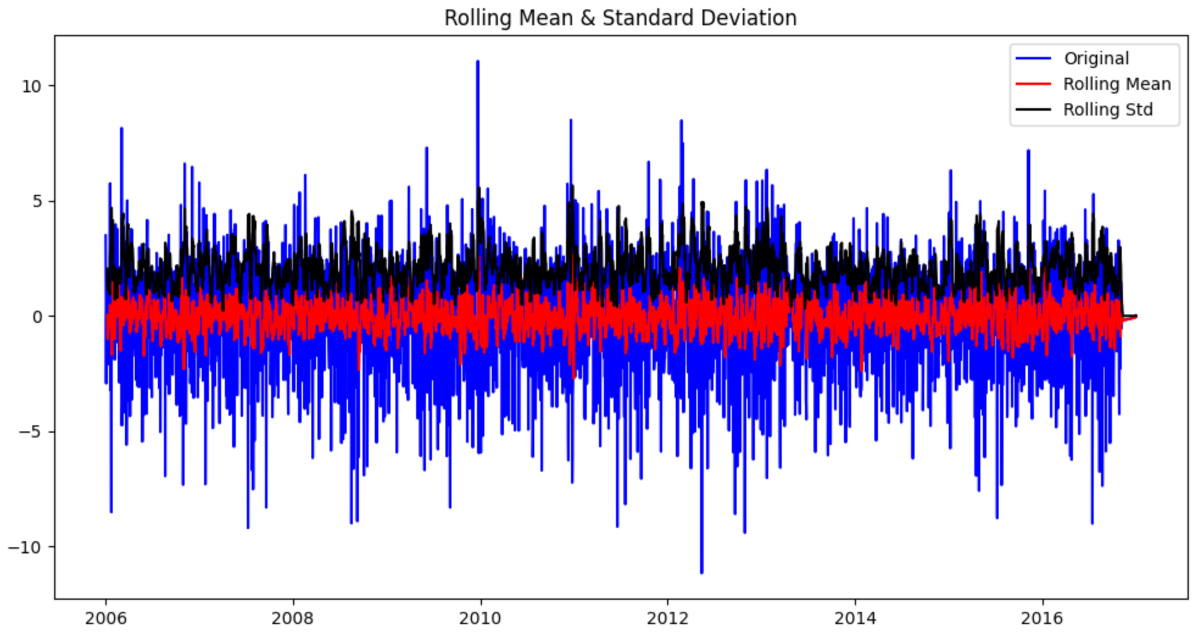
Overall, the code offers a thorough and useful illustration of how to apply ARIMA models to forecast time series data, in this case temperature. It serves as an example of key procedures such as data preparation, stationarity testing, model fitting, and forecasting, as well as a suitable starting point for additional research and model enhancements.



Temperature (C)
Temperature (C)
Temperature (C)
30
20
10
0
−10
2006
2008
2010
2012
2014
2016
Date



Rolling Mean & Standard Deviation
30
20
10
0
−10
2006
2008
2010
2012
2014
2016
Original
Rolling Mean
Rolling Std



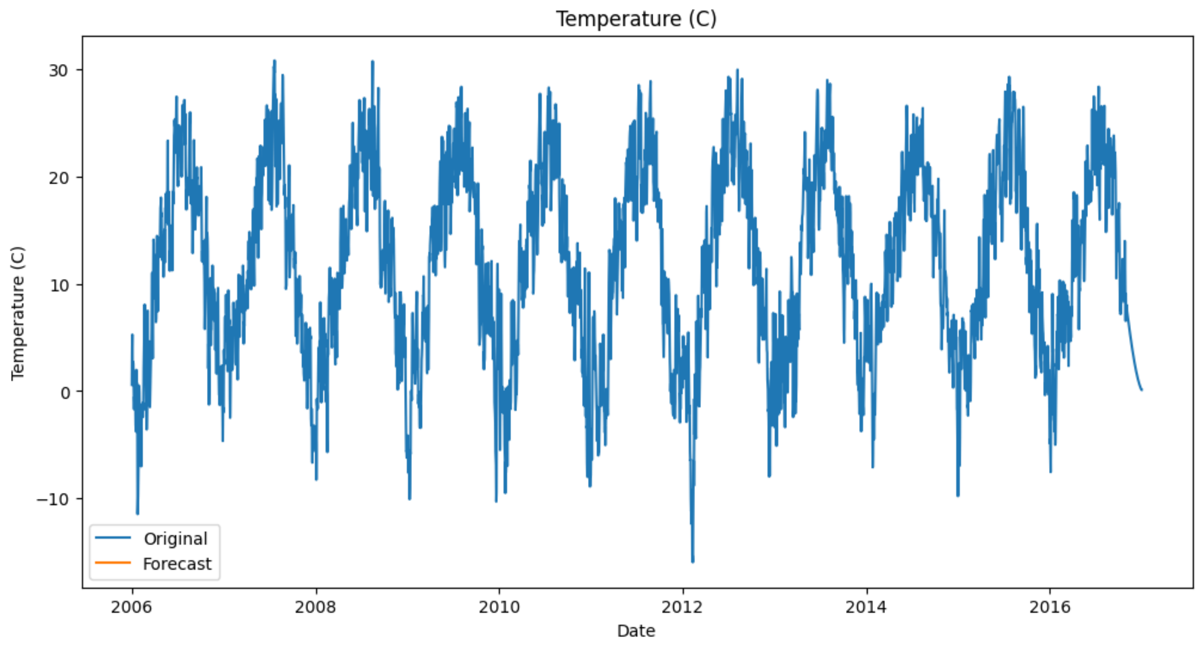
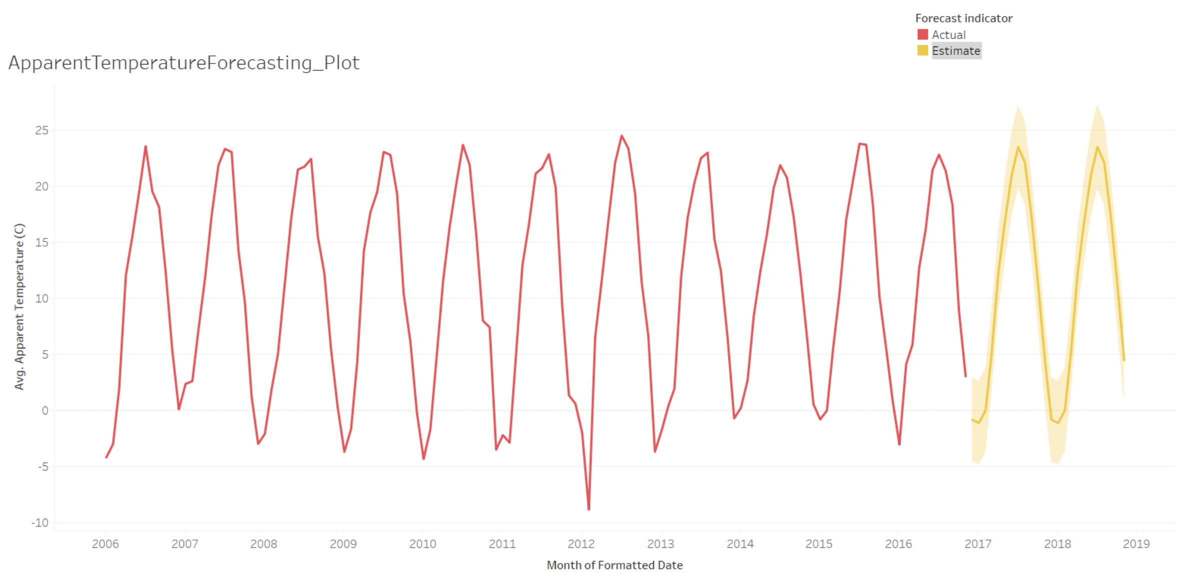
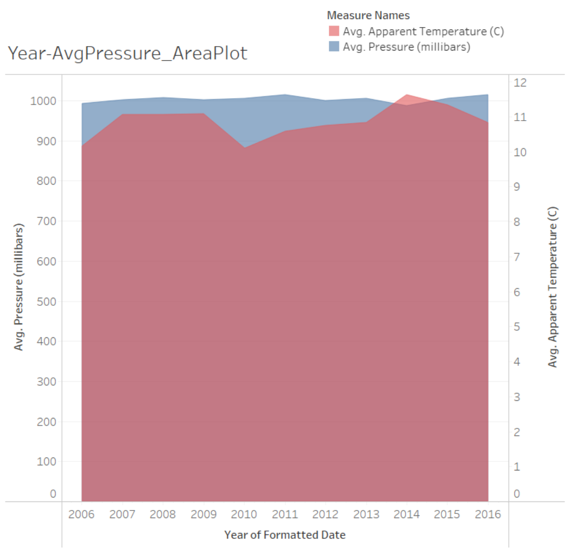
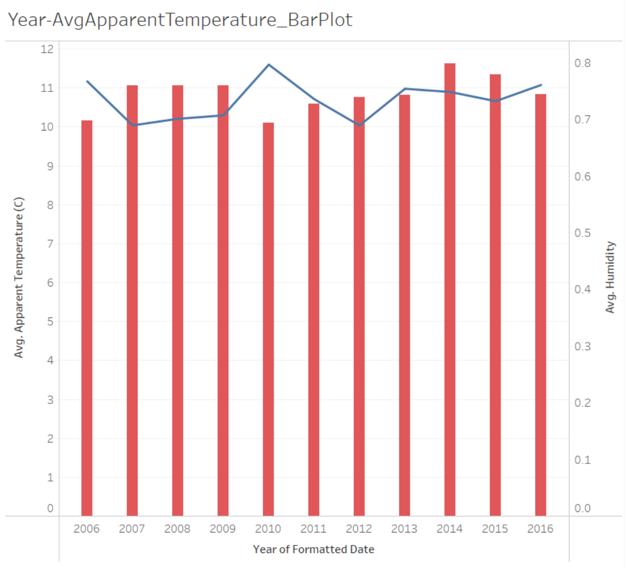
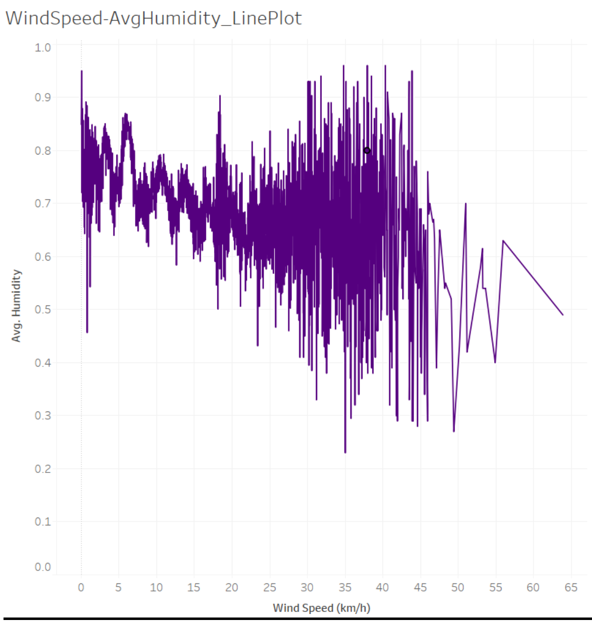


Tableau Dashboards:

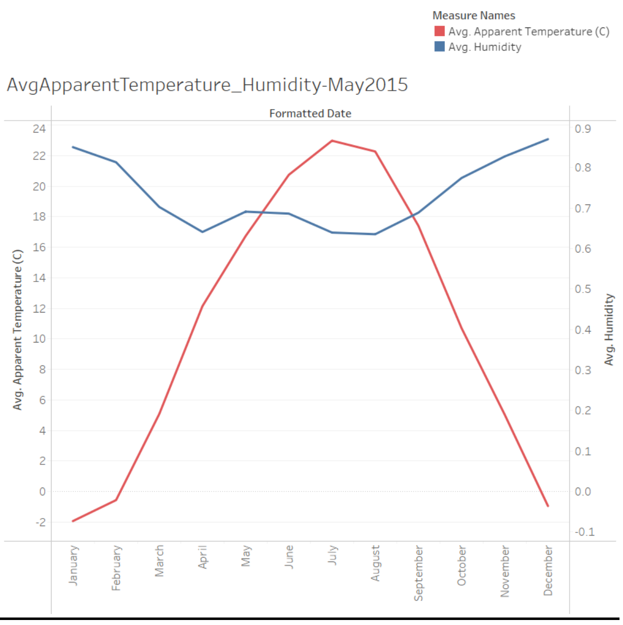

Forecast indicator
Actual
Estimate
ApparentTemperatureForecasting_Plot
Avg. Apparent Temperature (C)
25
20
15
10
5
0
-5
-10
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Month of Formatted Date


Year-AvgPressure_AreaPlot
Measure Names
Avg. Apparent Temperature (C)
Avg. Pressure (millibars)
Avg. Pressure (millibars)
Avg. Apparent Temperature (C)
Year of Formatted Date

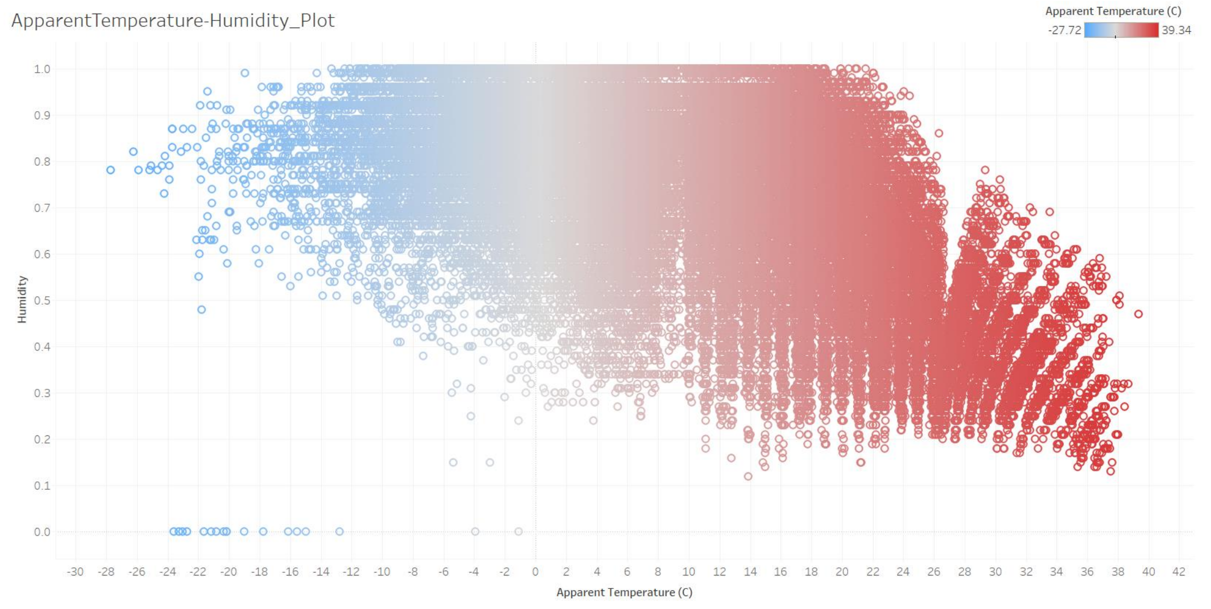

Year-AvgApparentTemperature_BarPlot
Avg. Apparent Temperature (C)
Avg. Humidity
Year of Formatted Date



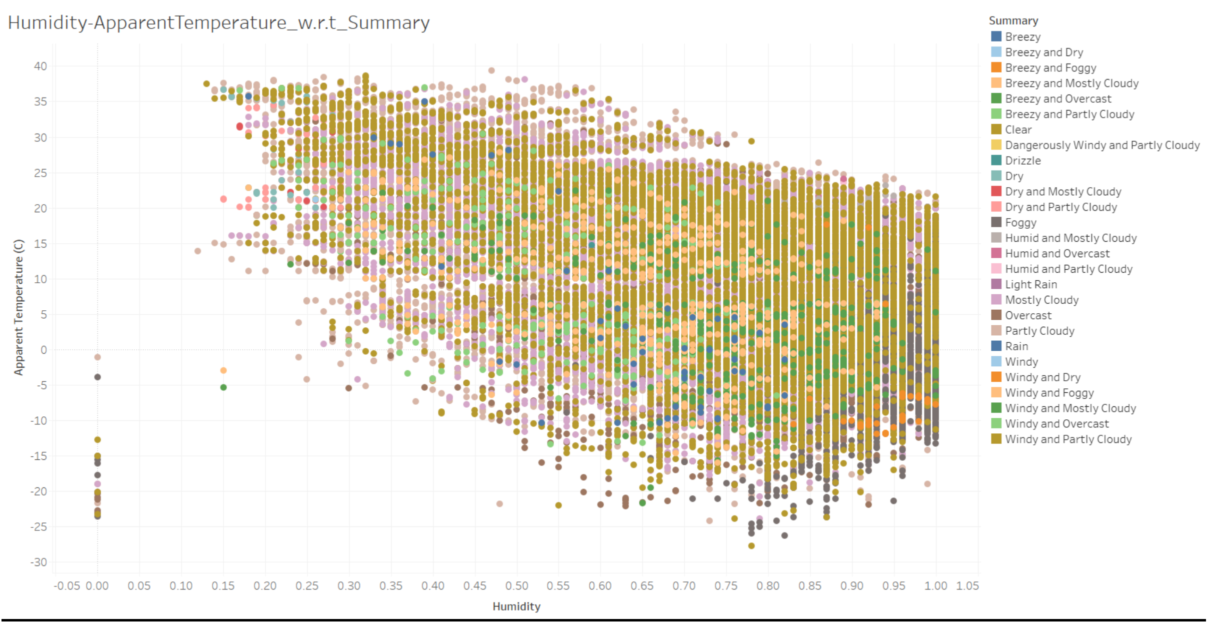
WindSpeed-AvgHumidity_LinePlot
Avg. Humidity
Wind Speed (km/h)



Measure Names
Avg. Apparent Temperature (C)
Avg. Humidity
AvgApparentTemperature_Humidity-May2015
Formatted Date
Avg. Apparent Temperature (C)
Avg. Humidity
January
February
March
April
May
June
July
August
September
October
November
December



ApparentTemperature-Humidity_Plot
Apparent Temperature (C)
-27.72
39.34
Humidity
Apparent Temperature (C)



Conclusion:

In conclusion, the code demonstrated the use of an ARIMA model to anticipate temperature data for the following two years and provided a thorough analysis and visualisation of meteorological data. To remove extraneous columns and fill in missing values, the data was initially preprocessed. The association between elements like humidity, wind speed, and temperature was then determined using monthly averages and data visualisation.

The time series data were then subjected to the ARIMA model, which was fitted using an order of (1,1,1) after differencing was performed to confirm the data's stationarity. The forecasted temperature values were calculated by multiplying the most recent temperature value by the cumulative sum of the forecasted temperature values.

The analysis offers a helpful place to start for more research and model enhancements. To comprehend the discrepancy between the predicted and actual temperature readings, multiple measures, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE), can be used to measure the model's accuracy. To find the optimal order for forecasting the temperature data, various orders for the ARIMA model could also be attempted and contrasted.

Overall, the code offers a superb illustration of how to reliably anticipate time series data using a variety of techniques, including data pre-treatment, visualization, and modelling.

References:

[1]. Meenal, Rajasekaran & Michael, Prawin & Pamela, D. & Rajasekaran, Ekambaram. (2021). Weather prediction using random forest machine learning model. Indonesian Journal of Electrical Engineering and Computer Science. 22. 1208. 10.11591/ijeecs.v22.i2.pp1208-1215.

[2]. Ho, C.-Y.; Cheng, K.-S.; Ang, C.-H. Utilizing the Random Forest Method for Short-Term Wind Speed Forecasting in the Coastal Area of Central Taiwan. Energies 2023, 16, 1374.

[3]. ANOKYE R., ACHEAMPONG E., OWUSU I., OBENG E.I. 2018. Time series analysis of malaria in Kumasi: Using ARIMA models to forecast future incidence. Cogent Social Sciences. Vol. 4(1) p. 1461544.

[4]. TADESSE K.B., DINKA M.O. 2017. Application of SARIMA model to forecasting monthly flows in Waterval River, South Africa. Journal of Water and Land Development. No. 35 p. 229–236. DOI 10.1515/jwld-2017-0088.

[5]. Murat, Małgorzata & Malinowska, Iwona & Gos, Magdalena & Krzyszczak, Jaromir. (2018). Forecasting daily meteorological time series using ARIMA and regression models. International Agrophysics. 32. 10.1515/intag-2017-0007.

[6]. Tektas, Mehmet. (2010). Weather Forecasting Using ANFIS and ARIMA MODELS: Case study for İstanbul. Environmental Research Engineering and Management. 51. 5-10. 10.5755/j01.erem.51.1.58.

[7]. Dimri, T., Ahmad, S. & Sharif, M. Time series analysis of climate variables using seasonal ARIMA approach. *J Earth Syst Sci* **129**, 149 (2020).

[8]. Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*.

[9]. Chen, Peng, et al. "Time series forecasting of temperatures using SARIMA: An example from Nanjing." *IOP conference series: materials science and engineering*. Vol. 394. No. 5. IOP Publishing, 2018.

[10]. Shivhare, Nikita & Dwivedi, S.B. & RAHUL, ATUL & Dikshit, Prabhat. (2019). ARIMA based daily weather forecasting tool: A case study for Varanasi. Mausam. 70. 133-140. 10.54302/mausam.v70i1.179.

Appendix:

code:

Data analysis:

Info about models—(algo/ pseudocodes) for the below

```
import pandas as pd

df = pd.read_csv("D:/college works/6th Semester/Data Visualization/Meteorological_data_analysis_and_forecasting/dataset/weatherHistory.csv")

df.head(50)

df.describe()

df.info()

df.sort_values(by='Formatted Date')

# removing the columns which are irrelevant for the analysis


df.drop(['Daily Summary','Loud Cover'],axis=1,inplace=True)

df

# analyzing the data for the missing values if any


df.columns[df.isnull().any()] # This Shows That 'Precip Type' Column has Missing Values

df.info() # This Also Shows That 'Precip Type' Column has Missing Values (95936 instead of 96453)

df[df.isnull().any(axis=1)] # This Shows All Entries with Missing Values

# Filling The Missing Data with "Not Defined" Keyword, in-order to use this 'Precip Type' Column for Later Analysis.


df.fillna("Not Defined", inplace=True)

df

df[df.isnull().any(axis=1)] # This Shows that there are no longer missing values in any column of our DataFrame.

df.info() # This Also Shows that there are no longer missing values in any column of our DataFrame.
df.shape # This Shows the Current Status or "Shape" of our DataFame.

df[['Formatted Date']] # Initial State of Dates in DataFrame

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

df

df_2=df.set_index('Formatted Date')

df_2 = df_2.resample('M').mean()

df_2


# This is the breakdown of Data With Respect to Month. It Accounts for 11 Years and 1 Month
of Data. [Extra Month of 2005]

df_2.info()

df_2 # This is the Data we have left to apply Visualization, Resampled into Months.

# We used Seaborn Library for Visualization, Because It has more customization as compared
to Matplotlib


import matplotlib.pyplot as plt

import seaborn as sns


plt.figure(figsize=(12,8))

g = sns.relplot(x='Apparent Temperature (C)', y='Humidity', data=df, kind="scatter",
color="indigo")

g.fig.suptitle("Apparent Temperature (C) vs. Humidity",y=1.1)

plt.show()


from IPython.core.display import HTML

HTML("""

<style>

.output_png {

    display: table-cell;

    text-align: center;
    vertical-align: middle;
}
</style>
""")
```

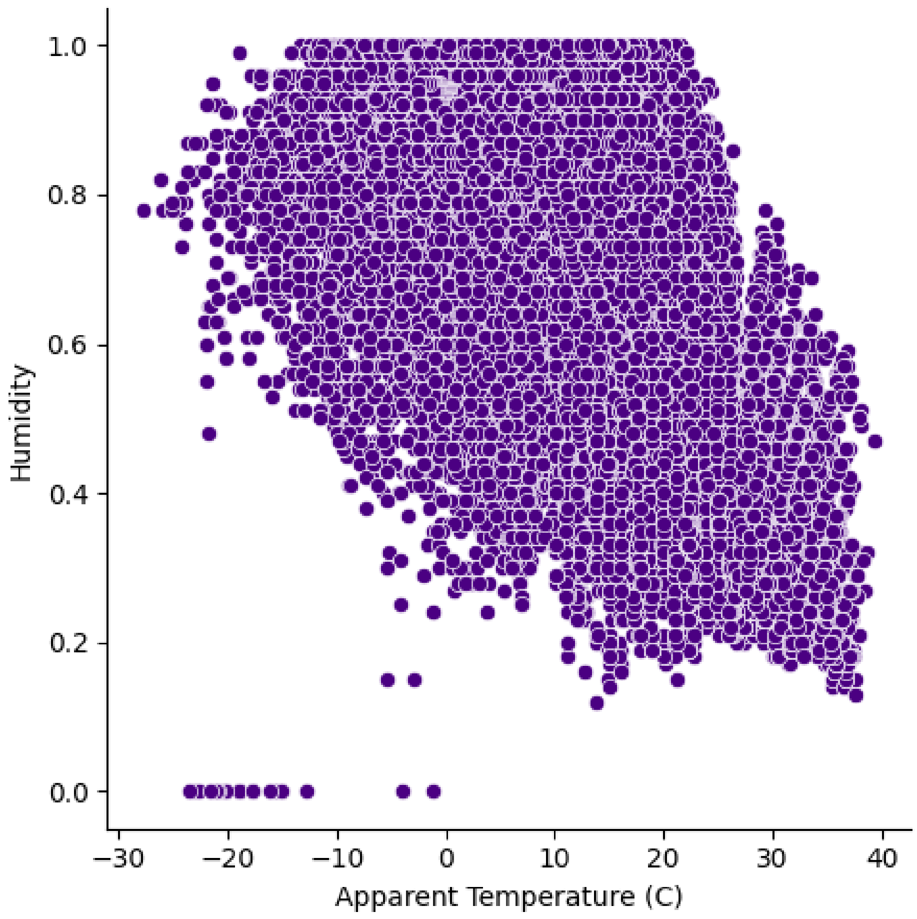
# According to Scientific Research, The Higher wind speed causes minimum evaporation of water, and low humidity, or vice versa. We can see the trend being proven in this plot below.

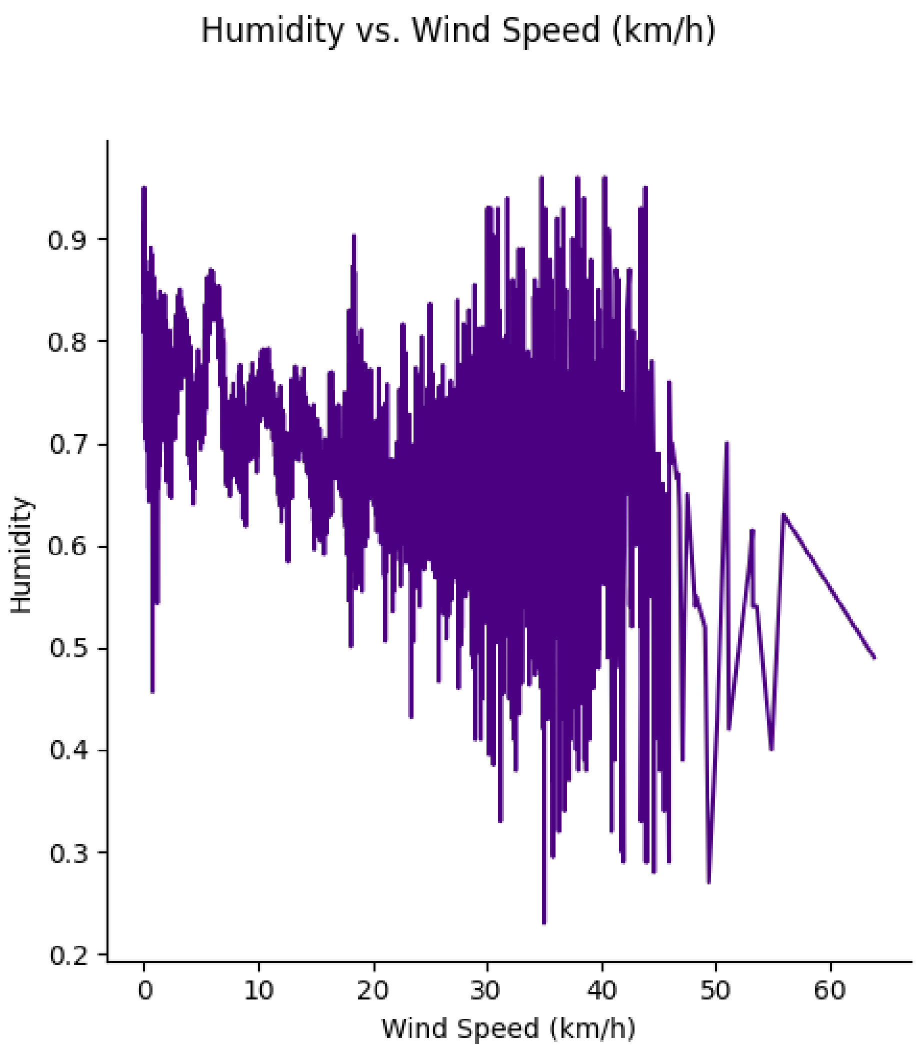
```
plt.figure(figsize=(12,8))
g = sns.relplot(x='Wind Speed (km/h)',y='Humidity', data=df, kind="line", color="indigo", ci=None)
g.fig.suptitle("Humidity vs. Wind Speed (km/h)",y=1.1)
plt.show()
g = sns.pairplot(df, vars=['Humidity', 'Apparent Temperature (C)'], hue='Summary')
g.fig.suptitle("Apparent Temperature (C) vs. Humidity w.r.t Summary",y=1.2)
plt.show()
plt.figure(figsize=(16,6))
g = sns.relplot(x='Apparent Temperature (C)',y='Humidity',color='purple',hue="Summary",data=df, col="Summary", col_wrap=3)
g.fig.suptitle("Apparent Temperature (C) vs. Humidity w.r.t Summary",y=1.005)
plt.show()
plt.figure(figsize=(16,8))
plt.title("Apparent Temperature and Humidity w.r.t Year")
plt.plot(df_2["Humidity"],label='Humidity',color='orange', marker='.')
plt.plot(df_2['Apparent Temperature (C)'],label='Apparent Temperature (C)', color='indigo', marker='.')
plt.xlabel("Year",fontsize="10")
plt.legend(loc=4,fontsize=10)
plt.show()
Month_wise = df_2[df_2.index.month==4]
Month_wise
fig, axes=plt.subplots(figsize=(20,8))
```

```
plt.title("Apparent Temperature and Humidity for the Month of April")

plt.plot(Month_wise['Humidity'],label='Humidity',color='orange',marker='o')

plt.plot(Month_wise['Apparent Temperature (C)'],label='Apparent Temperature
(C)',color='red',marker='o')

plt.xlabel('April Month',fontsize=10,color='Blue')

plt.legend(loc=0,fontsize=10)

plt.show()

jan=df_2[df_2.index.month==1]

feb=df_2[df_2.index.month==2]

march= df_2[df_2.index.month==3]

April=df_2[df_2.index.month==4]

May=df_2[df_2.index.month==5]

June=df_2[df_2.index.month==6]

July=df_2[df_2.index.month==7]

Aug=df_2[df_2.index.month==8]

sept=df_2[df_2.index.month==9]

octo=df_2[df_2.index.month==10]

nov=df_2[df_2.index.month==11]

dec=df_2[df_2.index.month==12]

plt.figure(figsize=(20,8))

plt.title("Plot of Humidity-Variation for All 12 months of 11 Years")

plt.plot(jan['Humidity'],label='January',color='red',marker='o')

plt.plot(feb['Humidity'],label='February',color='purple',marker='o')

plt.plot(march['Humidity'],label='March',color='yellow',marker='o')

plt.plot(April['Humidity'],label='April',color='black',marker='o')

plt.plot(May['Humidity'],label='May',color='orange',marker='o')

plt.plot(June['Humidity'],label='June',color='blue',marker='o')

plt.plot(July['Humidity'],label='July',color='green',marker='o')

plt.plot(Aug['Humidity'],label='Aug',color='pink',marker='o')

plt.plot(sept['Humidity'],label='September',color='brown',marker='o')
plt.plot(octo['Humidity'],label='October',color='white',marker='o')
plt.plot(nov['Humidity'],label='November',color='grey',marker='o')
plt.plot(dec['Humidity'],label='December',color='silver',marker='o')

plt.legend(loc=0,fontsize=10)
plt.figure(figsize=(24,8))
plt.title("Plot of Apparent Temperature (C) - Variation for All 12 months of 11 Years")
plt.plot(jan['Apparent Temperature (C)'],label='January',color='red',marker='o')
plt.plot(feb['Apparent Temperature (C)'],label='February',color='purple',marker='o')
plt.plot(march['Apparent Temperature (C)'],label='March',color='yellow',marker='o')
plt.plot(April['Apparent Temperature (C)'],label='April',color='black',marker='o')
plt.plot(May['Apparent Temperature (C)'],label='May',color='orange',marker='o')
plt.plot(June['Apparent Temperature (C)'],label='June',color='blue',marker='o')
plt.plot(July['Apparent Temperature (C)'],label='July',color='green',marker='o')
plt.plot(Aug['Apparent Temperature (C)'],label='Aug',color='pink',marker='o')
plt.plot(sept['Apparent Temperature (C)'],label='September',color='brown',marker='o')
plt.plot(octo['Apparent Temperature (C)'],label='October',color='white',marker='o')
plt.plot(nov['Apparent Temperature (C)'],label='November',color='grey',marker='o')
plt.plot(dec['Apparent Temperature (C)'],label='December',color='silver',marker='o')

plt.legend(loc=0,fontsize=8)
sns.distplot(df.Humidity, color = 'red')
```

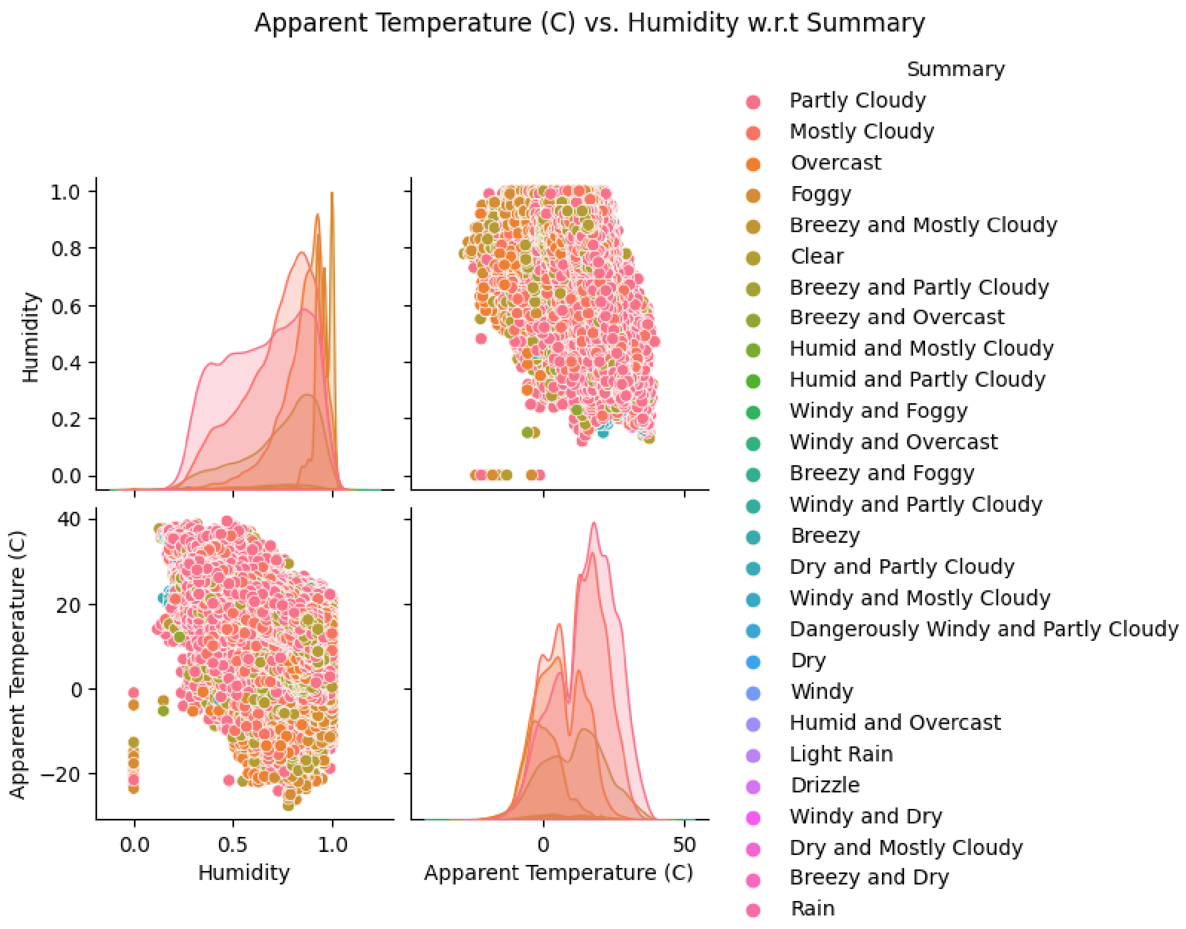
output:

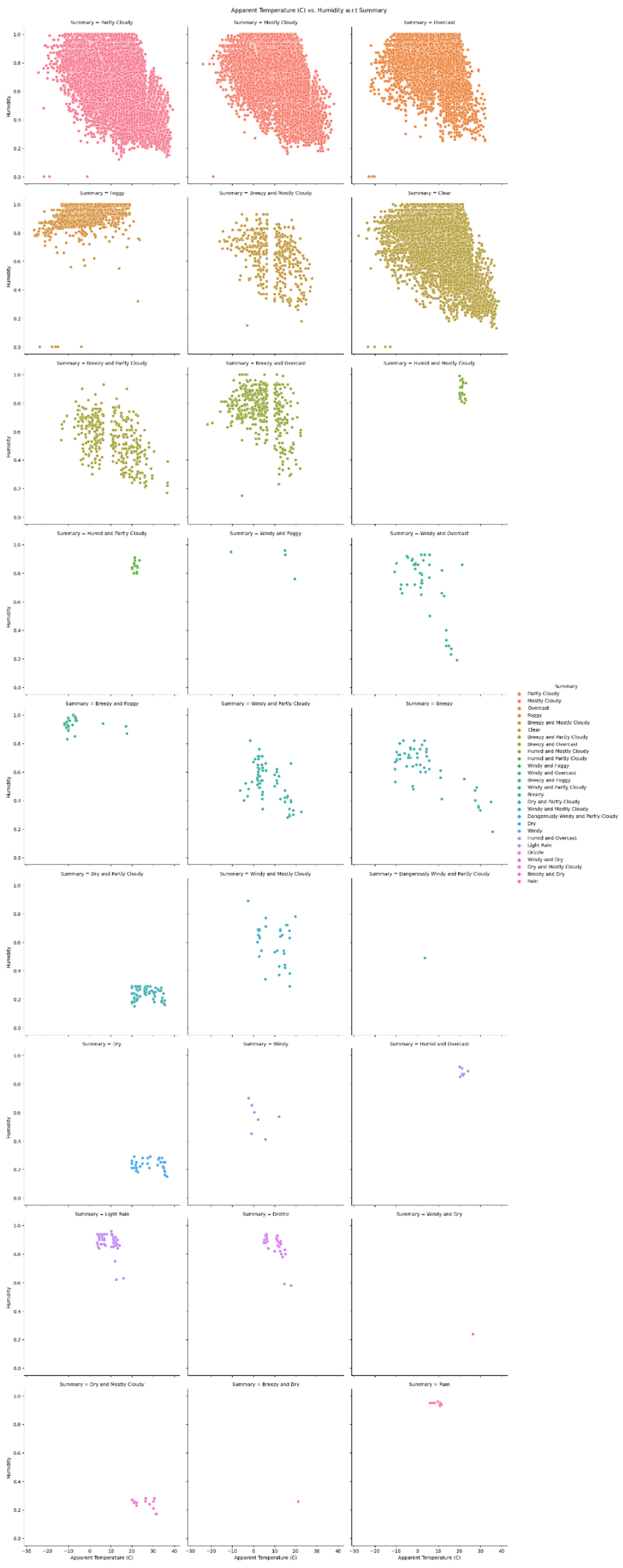
Apparent Temperature (C) vs. Humidity

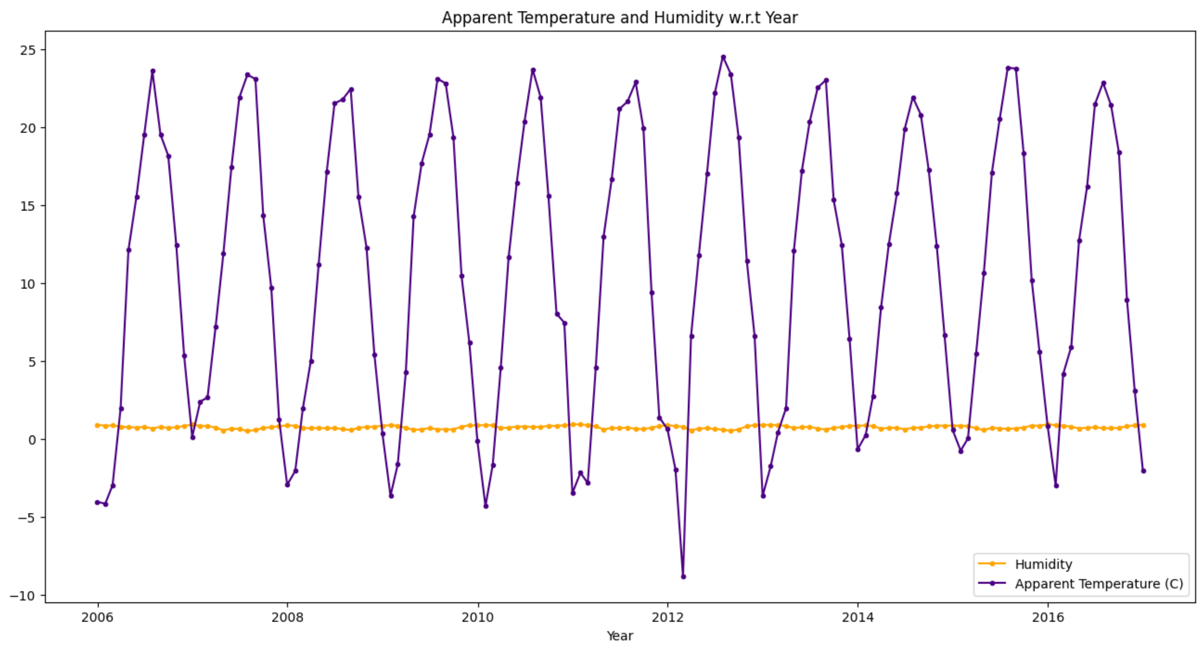


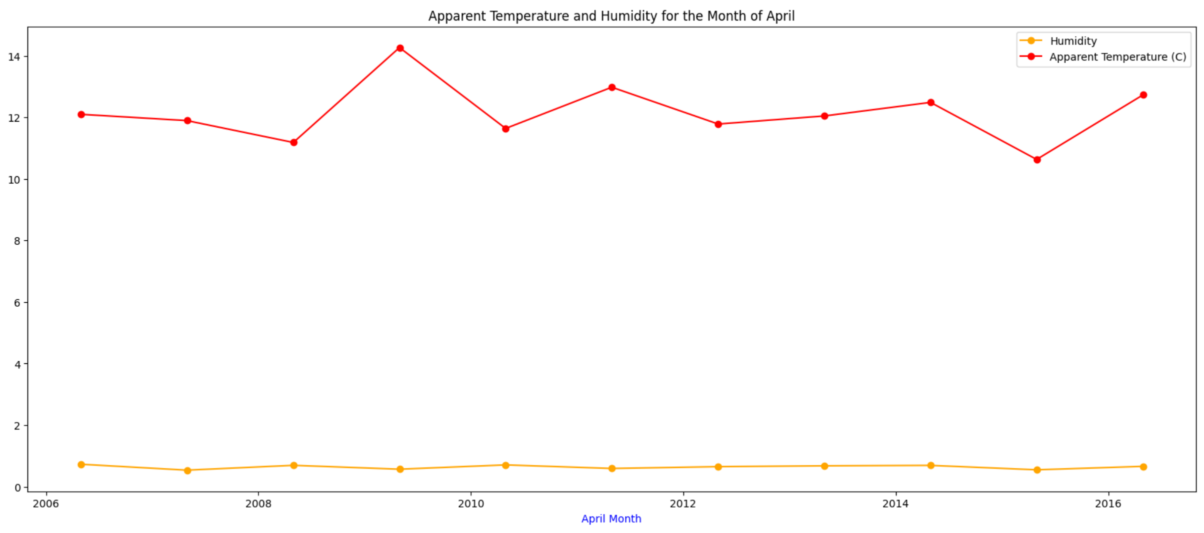


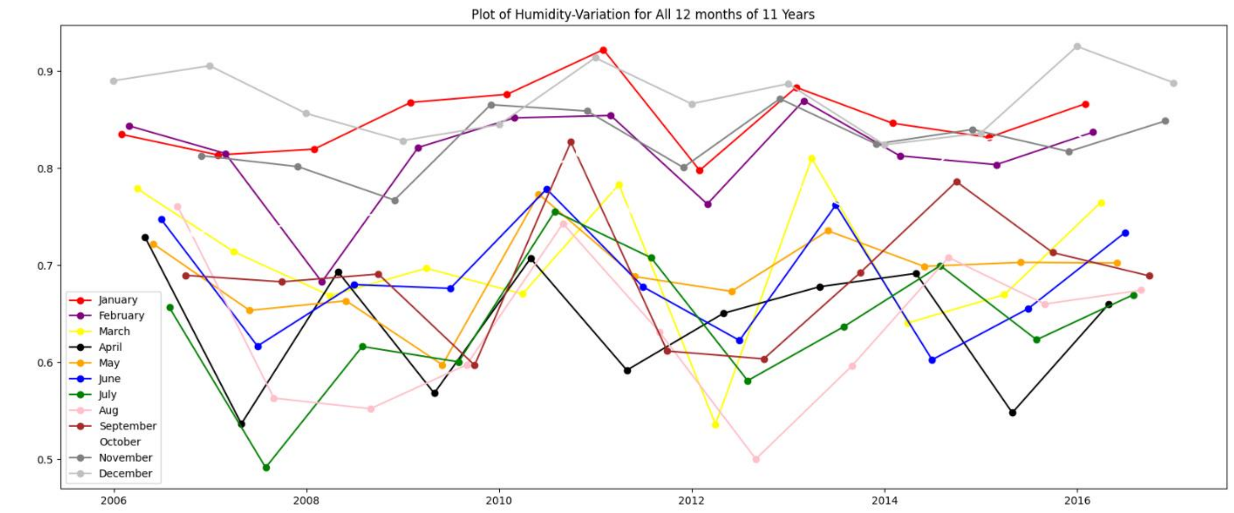
Humidity vs. Wind Speed (km/h)
Humidity
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Wind Speed (km/h)
0
10
20
30
40
50
60

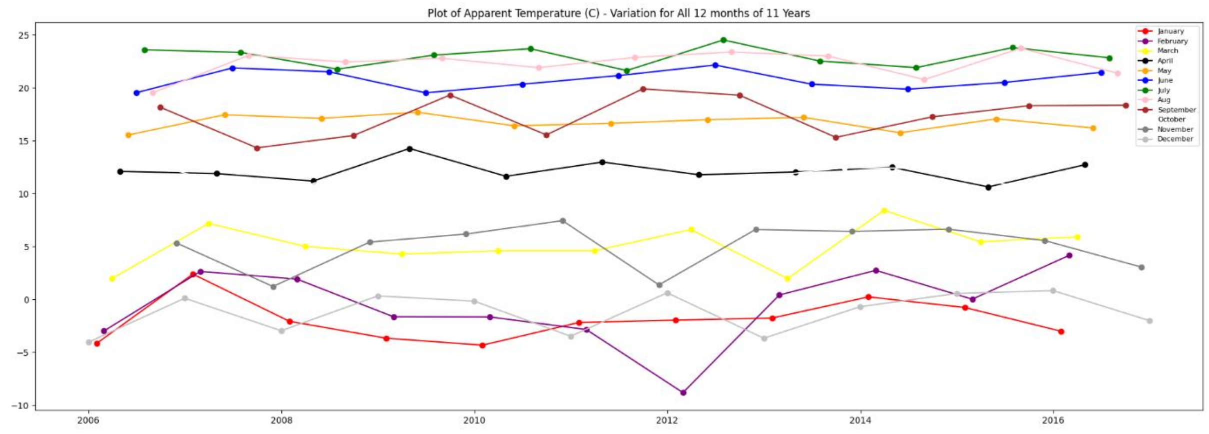

Apparent Temperature (C) vs. Humidity w.r.t Summary
Summary
Partly Cloudy
Mostly Cloudy
Overcast
Foggy
Breezy and Mostly Cloudy
Clear
Breezy and Partly Cloudy
Breezy and Overcast
Humid and Mostly Cloudy
Humid and Partly Cloudy
Windy and Foggy
Windy and Overcast
Breezy and Foggy
Windy and Partly Cloudy
Breezy
Dry and Partly Cloudy
Windy and Mostly Cloudy
Dangerously Windy and Partly Cloudy
Dry
Windy
Humid and Overcast
Light Rain
Drizzle
Windy and Dry
Dry and Mostly Cloudy
Breezy and Dry
Rain
Humidity
Apparent Temperature (C)
1.0
0.8
0.6
0.4
0.2
0.0
40
20
0
−20
0.0
0.5
1.0
0
50

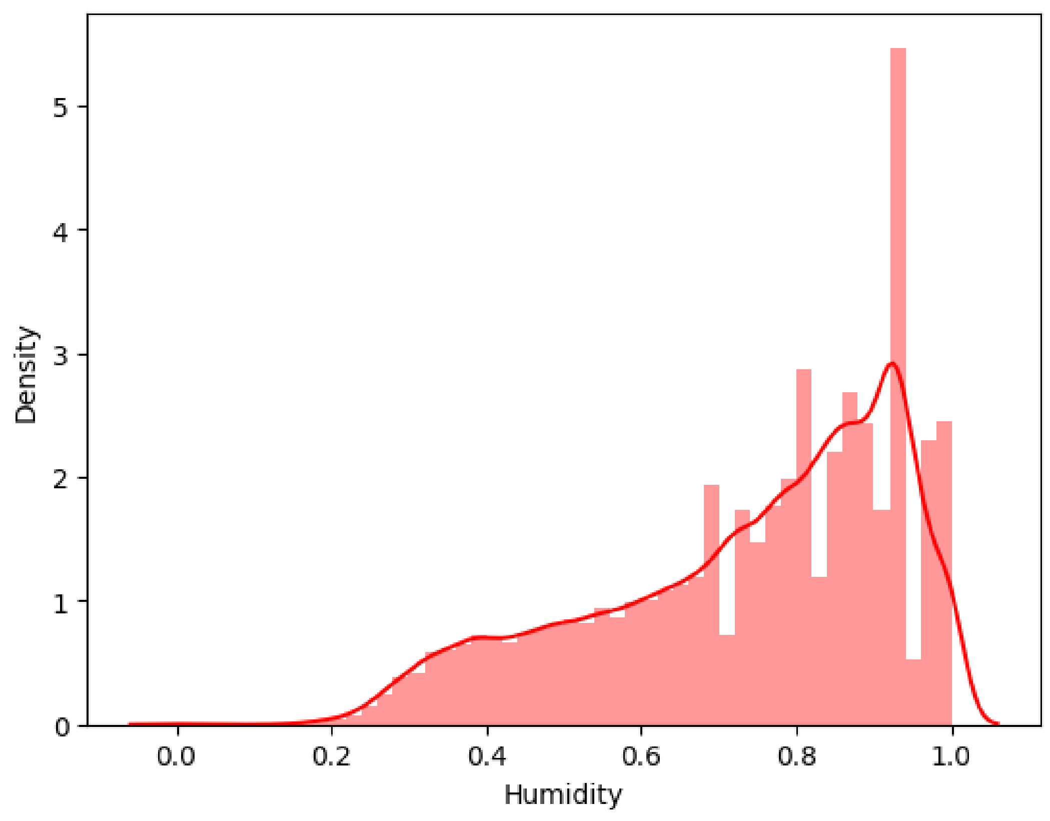

Apparent Temperature (C) vs. Humidity w.r.t Summary
Summary = Partly Cloudy
Summary = Mostly Cloudy
Summary = Overcast
Summary = Foggy
Summary = Breezy and Mostly Cloudy
Summary = Clear
Summary = Breezy and Partly Cloudy
Summary = Breezy and Overcast
Summary = Humid and Mostly Cloudy
Summary = Humid and Partly Cloudy
Summary = Windy and Foggy
Summary = Windy and Overcast
Summary = Breezy and Foggy
Summary = Windy and Partly Cloudy
Summary = Breezy
Summary = Dry and Partly Cloudy
Summary = Windy and Mostly Cloudy
Summary = Dangerously Windy and Partly Cloudy
Summary = Dry
Summary = Windy
Summary = Humid and Overcast
Summary = Light Rain
Summary = Drizzle
Summary = Windy and Dry
Summary = Dry and Mostly Cloudy
Summary = Breezy and Dry
Summary = Rain
Humidity
Apparent Temperature (C)
Summary
Partly Cloudy
Mostly Cloudy
Overcast
Foggy
Breezy and Mostly Cloudy
Clear
Breezy and Partly Cloudy
Breezy and Overcast
Humid and Mostly Cloudy
Humid and Partly Cloudy
Windy and Foggy
Windy and Overcast
Breezy and Foggy
Windy and Partly Cloudy
Breezy
Dry and Partly Cloudy
Windy and Mostly Cloudy
Dangerously Windy and Partly Cloudy
Dry
Windy
Humid and Overcast
Light Rain
Drizzle
Windy and Dry
Dry and Mostly Cloudy
Breezy and Dry
Rain


Apparent Temperature and Humidity w.r.t Year
Humidity
Apparent Temperature (C)
Year
25
20
15
10
5
0
−5
−10
2006
2008
2010
2012
2014
2016


Apparent Temperature and Humidity for the Month of April
Humidity
Apparent Temperature (C)
April Month
14
12
10
8
6
4
2
0
2006
2008
2010
2012
2014
2016


Plot of Humidity-Variation for All 12 months of 11 Years
January
February
March
April
May
June
July
Aug
September
October
November
December
0.9
0.8
0.7
0.6
0.5
2006
2008
2010
2012
2014
2016


Plot of Apparent Temperature (C) - Variation for All 12 months of 11 Years
January
February
March
April
May
June
July
Aug
September
October
November
December
25
20
15
10
5
0
−5
−10
2006
2008
2010
2012
2014
2016



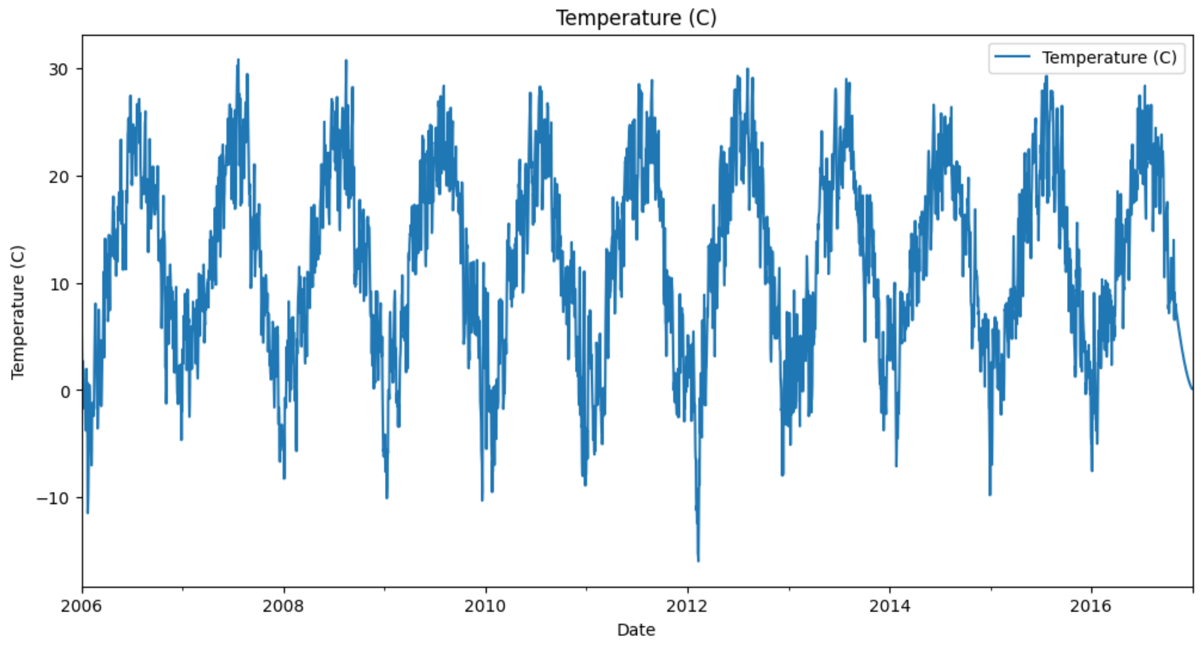
Forecasting:

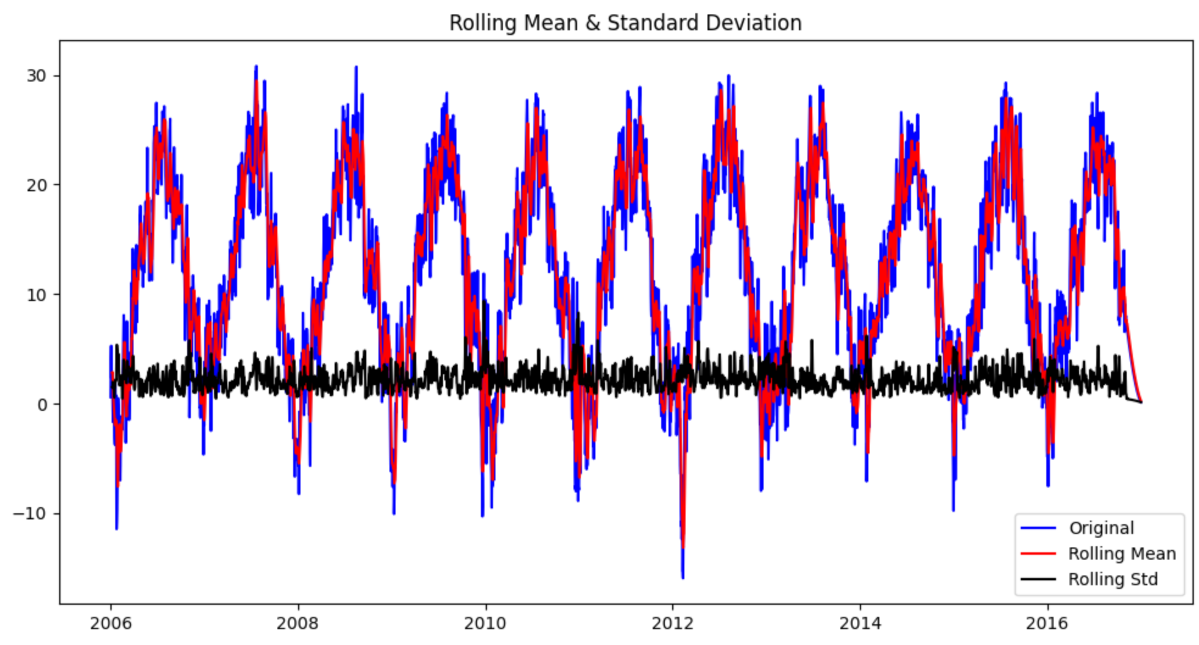
```
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
ts = df[['Formatted Date', 'Temperature (C)']].set_index('Formatted Date')
ts = ts.resample('D').mean().interpolate()
# Plot the time series
ts.plot(figsize=(12,6))
plt.title('Temperature (C)')
plt.xlabel('Date')
plt.ylabel('Temperature (C)')
plt.show()
```

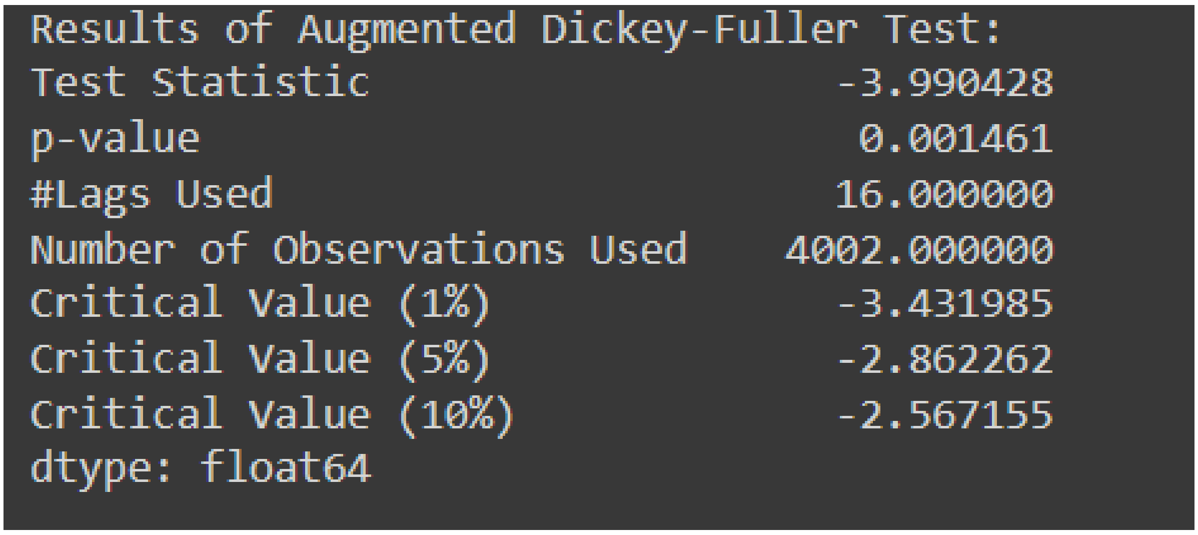
```
# Check the stationarity of the time series using the Augmented Dickey-Fuller test
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    # Determine rolling statistics
    rolmean = timeseries.rolling(window=7).mean()
    rolstd = timeseries.rolling(window=7).std()

    # Plot rolling statistics
    plt.figure(figsize=(12,6))
    orig = plt.plot(timeseries, color='blue', label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label='Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show()

    # Perform the Augmented Dickey-Fuller test
    print('Results of Augmented Dickey-Fuller Test:')
    adf_test = adfuller(timeseries, autolag='AIC')
    adf_output = pd.Series(adf_test[0:4], index=['Test Statistic','p-value','#Lags Used','Number
of Observations Used'])
    for key,value in adf_test[4].items():
        adf_output['Critical Value (%s)'%key] = value
    print(adf_output)

# Check the stationarity of the time series
test_stationarity(ts)
# The time series is not stationary because the rolling mean and standard deviation are not
constant over time and the p-value of the ADF test is greater than 0.05.
```

```
# To make the time series stationary, we can try differencing.


# Perform first-order differencing to make the time series stationary

ts_diff = ts.diff().dropna()


# Check the stationarity of the differenced time series

test_stationarity(ts_diff)

# The differenced time series appears to be stationary because the rolling mean and standard deviation are roughly constant over time and the p-value of the ADF test is less than 0.05.


# Fit an ARIMA model to the differenced time series

model = ARIMA(ts_diff, order=(1, 1, 1))

model_fit = model.fit()

print(model_fit.summary())

# Forecast the next 2 years days of the time series

forecast = model_fit.forecast(steps=730)[0]

forecast = pd.DataFrame(ts.iloc[-1], index=pd.date_range(start=ts.index[-1], periods=730, freq='D'), columns=['Forecast'])

forecast['Forecast'] = ts.iloc[-1] + forecast['Forecast'].cumsum()

print(ts)

#forecast['Forecast'] = ts.iloc[-1] + forecast['Forecast'].cumsum()

print(forecast)

# Reset the index of the original time series

ts_plot = ts.reset_index()

# Plot the original time series and the forecasted values

plt.figure(figsize=(12,6))

plt.plot(ts_plot['Formatted Date'], ts_plot['Temperature (C)'], label='Original')

plt.plot(forecast.index, forecast['Forecast'], label='Forecast')

plt.title('Temperature (C)')

plt.xlabel('Date')
plt.ylabel('Temperature (C)')

plt.legend(loc='best')

plt.show()
```

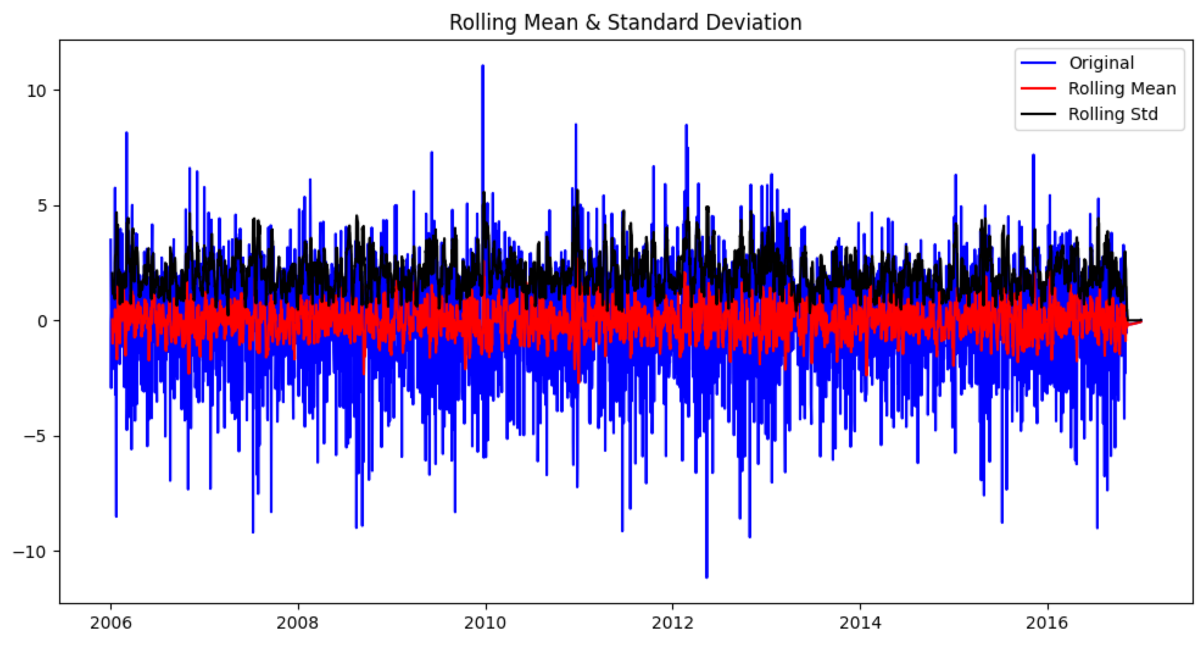
output:

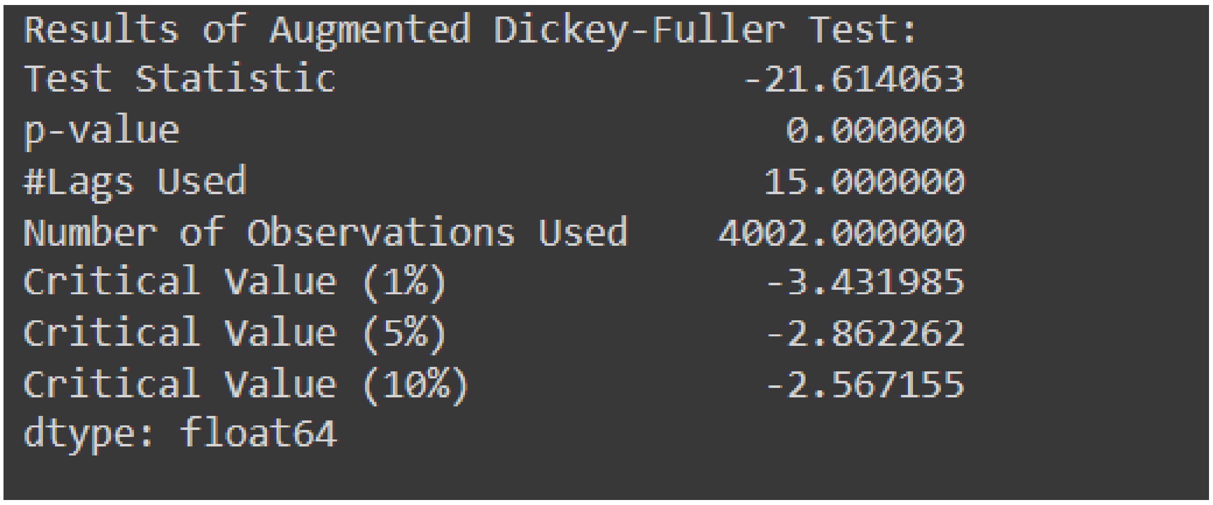




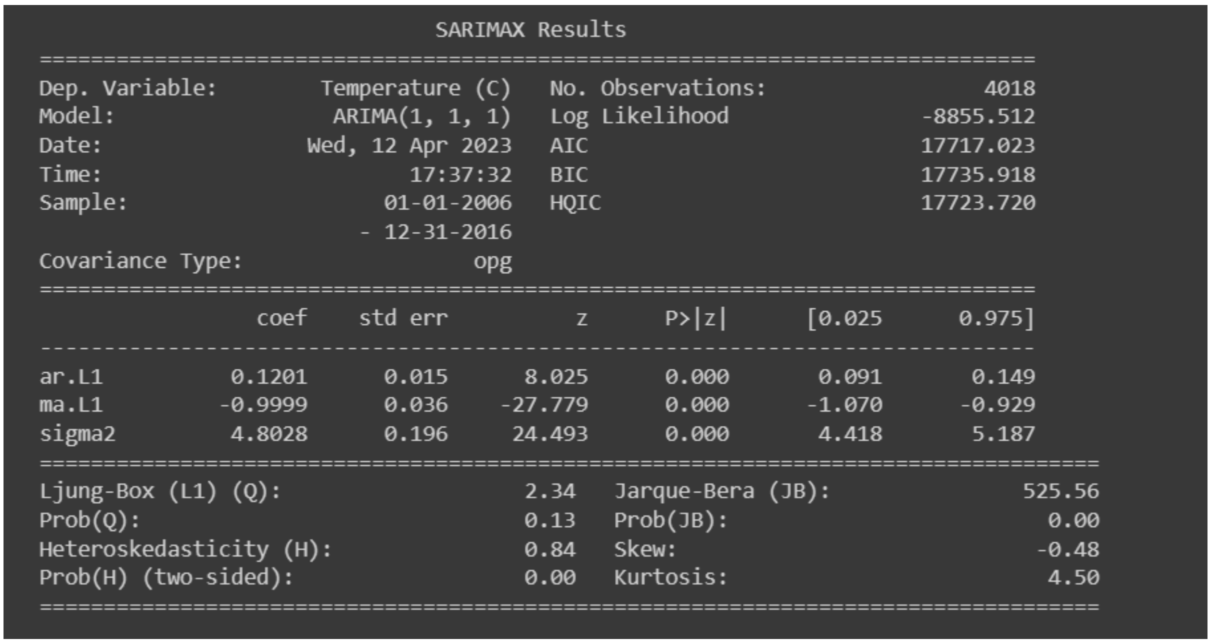


```
Results of Augmented Dickey-Fuller Test:
Test Statistic                    -3.990428
p-value                            0.001461
#Lags Used                        16.000000
Number of Observations Used     4002.000000
Critical Value (1%)               -3.431985
Critical Value (5%)               -2.862262
Critical Value (10%)              -2.567155
dtype: float64
```

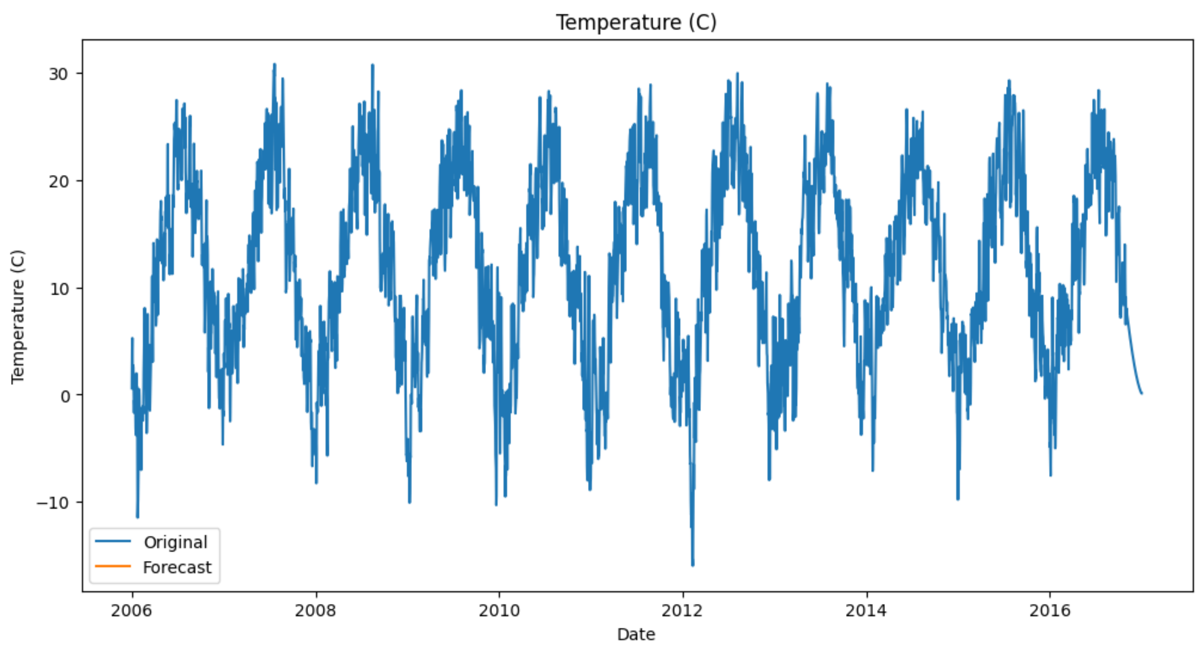




```
Results of Augmented Dickey-Fuller Test:
Test Statistic                  -21.614063
p-value                           0.000000
#Lags Used                       15.000000
Number of Observations Used    4002.000000
Critical Value (1%)              -3.431985
Critical Value (5%)              -2.862262
Critical Value (10%)             -2.567155
dtype: float64
```

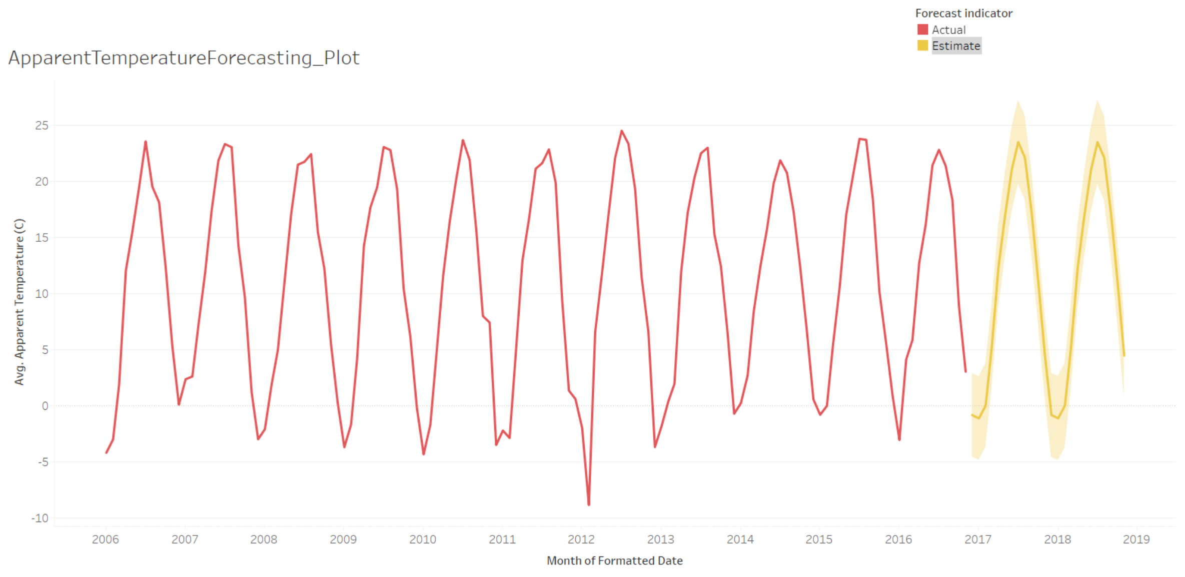


```
                               SARIMAX Results
==============================================================================
Dep. Variable:        Temperature (C)   No. Observations:                 4018
Model:                 ARIMA(1, 1, 1)   Log Likelihood               -8855.512
Date:                Wed, 12 Apr 2023   AIC                          17717.023
Time:                        17:37:32   BIC                          17735.918
Sample:                    01-01-2006   HQIC                         17723.720
                         - 12-31-2016
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1201      0.015      8.025      0.000       0.091       0.149
ma.L1         -0.9999      0.036    -27.779      0.000      -1.070      -0.929
sigma2         4.8028      0.196     24.493      0.000       4.418       5.187
===================================================================================
Ljung-Box (L1) (Q):                   2.34   Jarque-Bera (JB):               525.56
Prob(Q):                              0.13   Prob(JB):                         0.00
Heteroskedasticity (H):               0.84   Skew:                            -0.48
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.50
===================================================================================
```

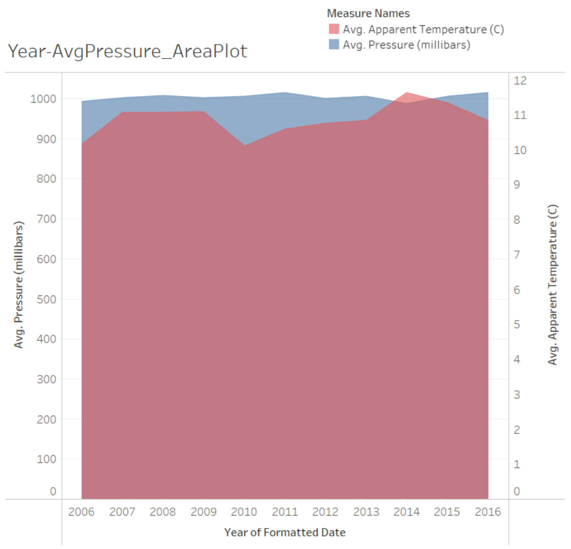


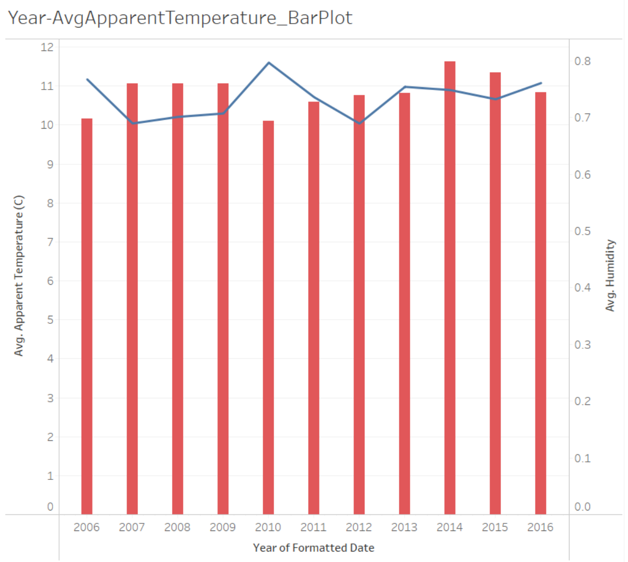
Temperature (C)
Temperature (C)
30
20
10
0
−10
Original
Forecast
2006
2008
2010
2012
2014
2016
Date

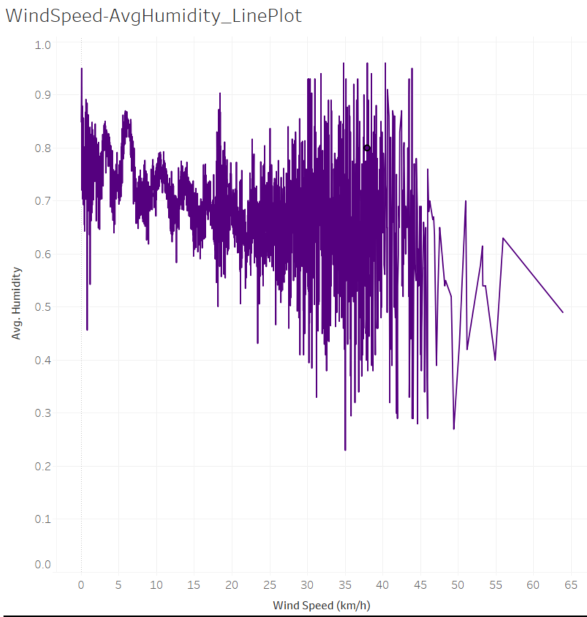
Tableau:

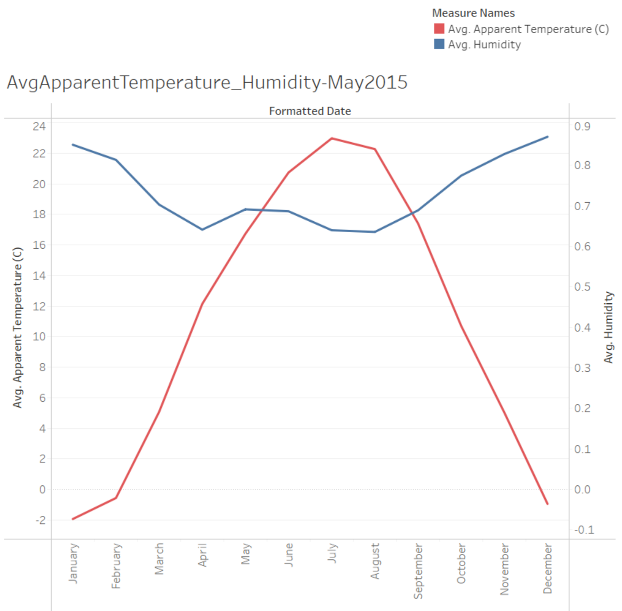


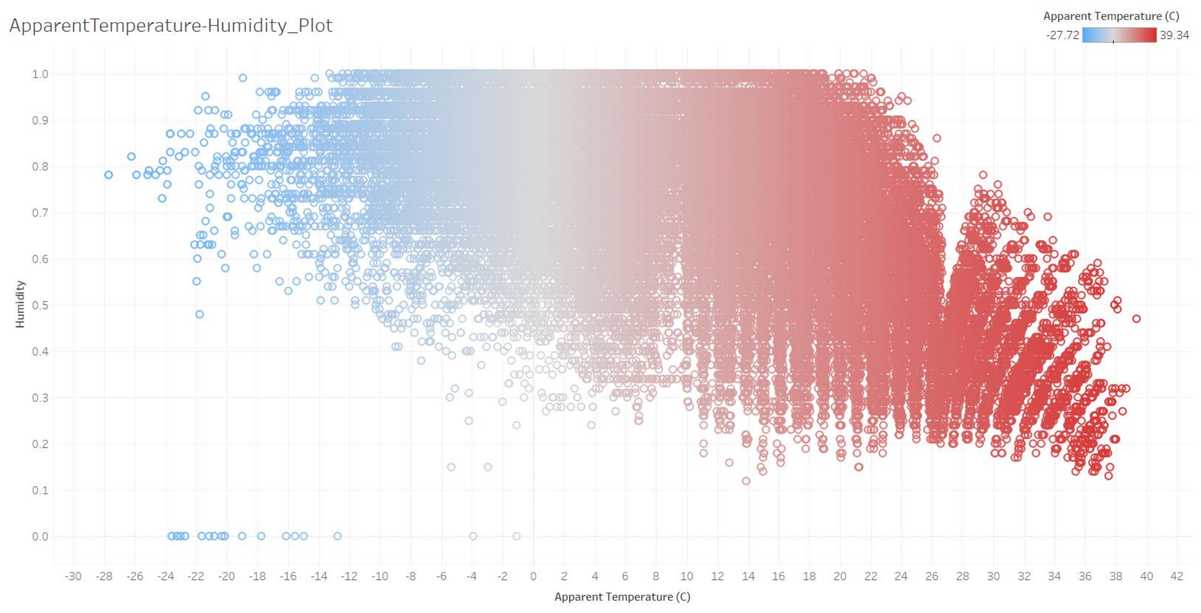
Forecast indicator
Actual
Estimate
ApparentTemperatureForecasting_Plot
Avg. Apparent Temperature (C)
25
20
15
10
5
0
-5
-10
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Month of Formatted Date


Measure Names
Avg. Apparent Temperature (C)
Avg. Pressure (millibars)
Year-AvgPressure_AreaPlot
Avg. Pressure (millibars)
Avg. Apparent Temperature (C)
Year of Formatted Date


Year-AvgApparentTemperature_BarPlot
Avg. Apparent Temperature (C)
Avg. Humidity
Year of Formatted Date


WindSpeed-AvgHumidity_LinePlot
Avg. Humidity
Wind Speed (km/h)


Measure Names
Avg. Apparent Temperature (C)
Avg. Humidity
AvgApparentTemperature_Humidity-May2015
Formatted Date
Avg. Apparent Temperature (C)
Avg. Humidity
January
February
March
April
May
June
July
August
September
October
November
December



Dataset link:

http://www.kaggle.com/datasets/muthuj7/weather-dataset

Github link:

Akshaykviit023/Meteorological-data-analysis-and-forecasting (github.com)