A

Report

On

Skill based Mini Project

On

"Weather Forecasting Using Time Series Analysis"

For the course of

Data Science (130515)

Submitted By:

SHASHANK CHANDRAVANSHI (0901EE211104)

Under the guidance of

Dr. Nikhil Paliwal

DEPARTMENT OF ELECTRICAL ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.), INDIA
माधव प्रौद्योगिकी एवं विज्ञान संस्थान, ग्वालियर (म.प्र.), भारत

A GOVT. AIDED UGC AUTONOMOUS INSTITUTE, AFFILIATED TO R.G.P.V. BHOPAL (M.P.), INDIA
NAAC ACCREDITED WITH A++ GRADE

Race Course Road, Gola Ka Mandir, Gwalior, M.P. 474005
Website: www.mitsgwalior.in

20 NOVEMBER 2023

CERTIFICATE

This is to certify that **SHASHANK CHANDRAVANSHI (0901EE211104)** studying in **MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE,** BATCH-2021-2025) have completed their Mini Skilled Based Project entitled **"Weather Forecasting Using Time Series Analysis"** at **MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE** under my supervision.

It is further certified that they had attended required number of practical classes at **MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE**,GWALIOR for the completion of their Mini Skilled Based Project during 5^{th} semester.

DR. NIKHIL PALIWAL

Project Supervisor

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this SKILLED BASED PROJECT entitled "**Weather Forecasting Using Time Series Analysis"** which is being submitted in the partial fulfillment of the requirement for the award of degree of Bachelor of Engineering in Electrical Engineering is an authentic record of our own work carried out under the guidance of DR. NIKHIL PALIWAL, Electrical Engineering Department.

The matter presented in this project has not been submitted elsewhere by us for the award of any other degree/diploma.

SHASHANK CHANDRAVANSHI

(0901EE211104)

Date: 20/11/2023

Place: Gwalior

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief.

Guided by

DR. NIKHIL PALIWAL
Department of Electrical Engineering
MITS, Gwalior

ACKNOWLEDGEMENT

Engineers in all disciplines must acquire knowledge of project making. Student, in particular, will find 'project making' as an integral part of their studies that will infuse the spirit of doing practical work in them.

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible whose constant guidance crowned our efforts with success.

We sincerely express our deep gratitude to the management of our college for giving us liberty to choose and to work on the most relevant project i.e. **"Weather Forecasting Using Time Series Analysis".** We are thankful to DR. NIKHIL PALIWAL for ensuring that we have a smooth environment at the college and lab. At the very outset we would like to offer our never ending thanks to our project supervisor DR. NIKHIL PALIWAL who helped us with our project from the beginning till the end. His continuous surveillance over our work allowed us to work more efficiently.

SHASHANK CHANDRAVANSHI

(0901EE211104)

ABSTRACT

Weather forecasting is a critical aspect of modern life, influencing everything from agriculture and transportation to disaster preparedness and public safety. Accurate predictions of future weather conditions can significantly impact decision-making, resource allocation, and overall quality of life. This project aims to improve weather forecasting by applying time series analysis techniques to historical weather data. The primary objective of this project is to develop a robust and accurate weather forecasting system that leverages time series analysis to model and predict various meteorological parameters, such as temperature, precipitation, humidity, and wind speed. To achieve this goal, the project employs a multi-step process, which includes data collection, preprocessing, model selection, and evaluation. The project utilizes historical weather data from various sources, including meteorological stations, satellites, and weather sensors. Data preprocessing techniques are applied to clean and prepare the data for analysis, including missing data imputation and feature engineering. Time series analysis methods, such as autoregressive models (ARIMA), seasonal decomposition, and machine learning algorithms, are then used to develop forecasting models.

Keywords: Weather forecasting, Time series analysis, Data preprocessing, Forecasting models, Meteorological parameters, Accuracy, Evaluation, Visualization. Weather prediction is the application of science and technology to predict the state of the atmosphere for a given location. Here this system will predict weather based on parameters such as temperature, humidity and wind. This system is a web application with effective graphical user interface. To predict the future's weather condition, the variation in the conditions in past years must be utilized. The probability that it will match within the span of adjacent fortnight of previous year is very high. We have proposed the use of linear regression for weather prediction system with parameters such as temperature, humidity and wind. It will predict weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

OBJECTIVES:

The objectives for a project focused on "Weather Forecasting Using Time Series Analysis" can be outlined as follows:

1. Data Collection:
- Gather historical weather data from reliable sources, including meteorological stations, satellites, and weather sensors.

2. Data Preprocessing:
- Clean and quality-check the collected data.
- Address missing data through imputation techniques.
- Perform data transformation and feature engineering to enhance the dataset's suitability for time series analysis.

3. Model Selection:
- Explore and select appropriate time series analysis techniques and forecasting models, such as ARIMA, Seasonal Decomposition, Exponential Smoothing, and machine learning algorithms.

4. Model Development:
- Build and train time series models using the selected techniques and algorithms.
- Optimize model hyperparameters to achieve accurate and reliable predictions.

5. Model Evaluation:
- Assess the performance of forecasting models using relevant metrics, such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and other appropriate evaluation measures.
- Implement cross-validation techniques to ensure robustness and generalizability of the models.

6. Visualization:
- Create visualizations to effectively communicate the forecast results and model diagnostics to end-users and stakeholders.
- Generate graphical representations of historical data, forecasts, and accuracy assessments.

7. Model Integration:
- Develop a user-friendly interface or system for accessing and utilizing the weather forecasting models.
- Ensure the integration of real-time or near-real-time data feeds for ongoing forecasting.

8. Accuracy Improvement:
- Continuously refine and enhance forecasting models to improve accuracy and reliability.
- Investigate and incorporate external data sources, such as climate patterns, to enhance predictions.

9. Scalability:
- Design the project with scalability in mind to accommodate the increasing volume of data and evolving modeling techniques.

10. Application and Usability:
- Identify specific applications and end-users for the weather forecasts (e.g., agriculture, transportation, disaster preparedness, energy management).
- Customize forecasting outputs and formats to meet the requirements of various stakeholders.

11. Documentation:
- Maintain comprehensive documentation of the project's methodology, data sources, and model details for transparency and future reference.

12. Research and Innovation:
- Stay updated with the latest developments in time series analysis and meteorology to incorporate innovative techniques and methods into the project.

13. Collaboration:
- Collaborate with meteorologists, domain experts, and other researchers to gather insights, validate models, and improve forecasting accuracy.

14. Validation and Verification:
- Compare forecasted results with ground truth observations to validate the accuracy of the models and make necessary adjustments.

15. Communication:
- Effectively communicate the project's findings, progress, and forecasts to the public, stakeholders, and decision-makers through reports, presentations, and accessible online platforms.

TABLE OF CONTENTS

Chapter 1
Introduction

1.1 Introduction to Weather Forecasting

Weather forecasting is the task of predicting the state of the atmosphere at a future time and a specified location. Traditionally, this has been done through physical simulations in which the atmosphere is modeled as a fluid. The present state of the atmosphere is sampled, and the future state is computed by numerically solving the equations of fluid dynamics and thermodynamics. However, the system of ordinary differential equations that govern this physical model is unstable under perturbations, and uncertainties in the initial measurements of the atmospheric conditions and an incomplete understanding of complex atmospheric processes restrict the extent of accurate weather forecasting to a 10 day period, beyond which weather forecasts are significantly unreliable. Machine learning, on the contrary, is relatively robust to perturbations and doesn't require a complete understanding of the physical processes that govern the atmosphere. Therefore, machine learning may represent a viable alternative to physical models in weather forecasting.

Machine learning is the ability of computer to learn without being explicitly programmed. It allows machines to find hidden patterns and insights. In supervised learning, we build a model based on labeled training data. The model is then used for mapping new examples. So, based on the observed weather patterns from the past, a model can be built and used to predict the weather.

This project work focuses on solving the weather prediction anomalies and in-efficiency based on linear regression algorithms and to formulate an efficient weather prediction model based on the linear regression algorithms.

Forecast of weather parameters using time series data:

The present study is undertaken to develop area specific weather forecasting models based on time series data for ANY COUNTRY OR STATE. The study was carried out by using time series secondary monthly weather data of 27 years (from 1981-82 to 2007-08). The trend analysis of weather parameters was done by Mann-Kendall test statistics. The methodologies adopted to forecast weather parameters were the winter's exponential smoothing model and Seasonal Autoregressive Integrated Moving Average . Comparative study has been carried out by using forecast error percentage and mean square error. The study showed that knowledge of this trend is likely to be helpful in planning and production of enterprises/crops. The study of forecast models revealed that SARIMA model is the most efficient model for forecasting of monthly maximum temperature, monthly minimum temperature and monthly humidity I. The Winter's model was found to be the most efficient model for forecasting Monthly Humidity II but no model was found to be appropriate to forecast monthly total rainfall.

1.2 Methodology used

In a developing country and an economy like India where major population is dependent on agriculture, weather conditions play an important and vital role in economic growth of the overall nation. So, weather prediction should be more precise and accurate. Weather parameters are collected from the open source . The data used in this project is of the years 2013-2019. The programming language used is 'Python'. Fig. 1.1 visualizes the system in the form of a block diagram.

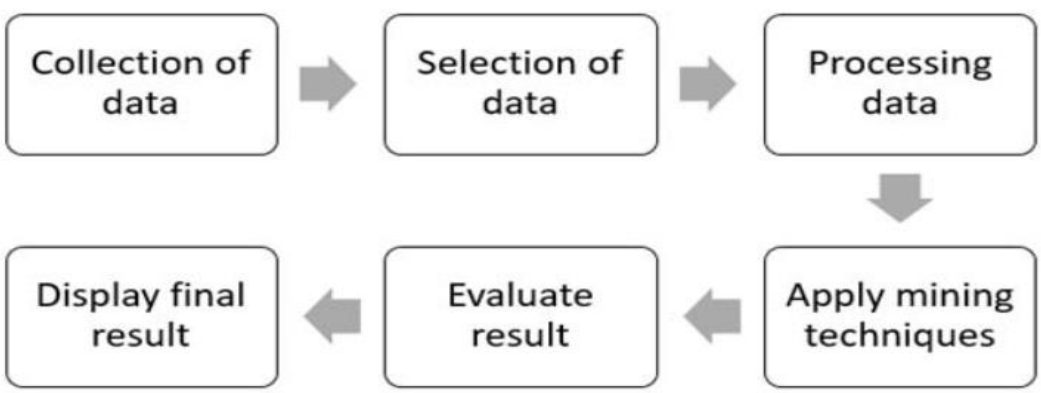


Fig 1.1 System Block Diagram

The weather is predicted using various indices like temperature, humidity and dew-point. Temperature is the measure of hotness or coldness, generally measured using thermometer. Units of temperature most frequently used are Celsius and Fahrenheit. We have used maximum and minimum temperature values along with normal temperature as different index values for prediction of the weather.

Humidity is the quantity of water vapor present in the atmosphere. It is a relative quantity.

Dew point is the temperature of the atmosphere (which varies according to pressure and humidity) below which water droplets begin to condense and dew is formed.

1.3 Technologies Used

1.3.1 Machine learning – Linear Regression

Linear regression is the most basic and frequently used predictive model for analysis. Regression estimates are generally used to describe the data and elucidate relationship between one or more independent and dependent variables. Linear regression finds the best-fit through the points, graphically. The best-fit line through the points is known as the regression line.

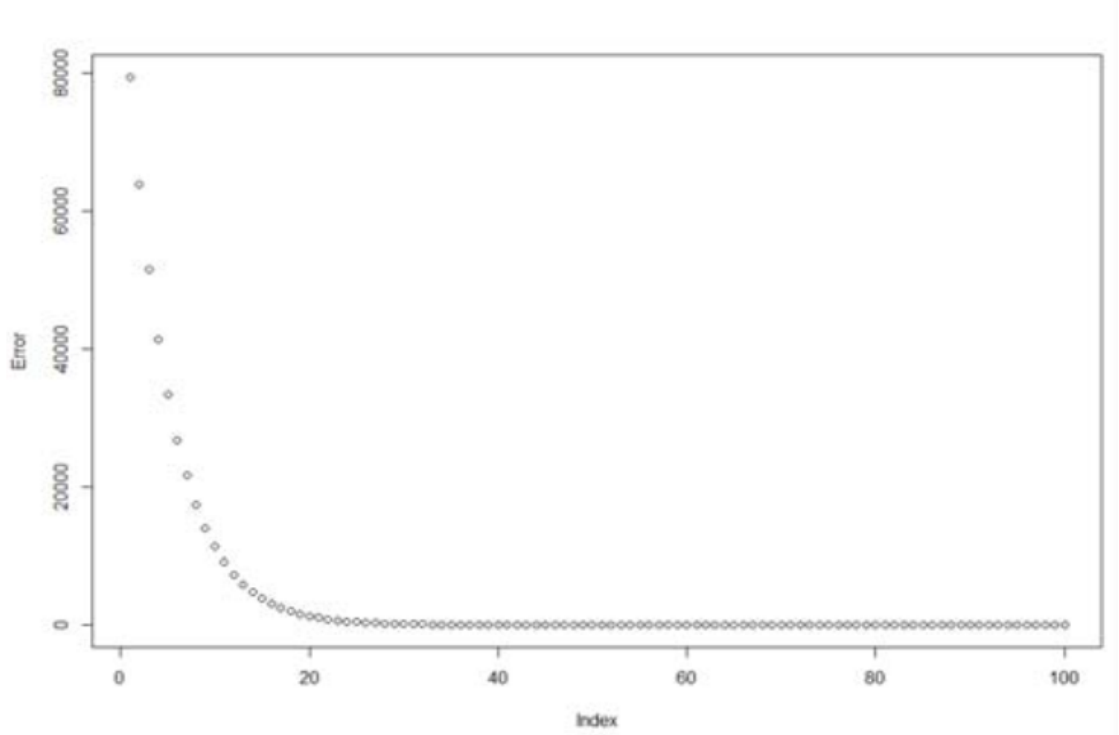


Fig. 1. Example of regression line

Fig. 1 is an example of the best-fit line. Here, the line can be straight or curved depending on the data. The best-fit line can also be a quadratic or polynomial which gives us better answer to our questions.

1.3.6 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed

1.3.7 Weather API

Weather APIs are Application Programming Interfaces that allow you to connect to large databases of weather forecast and historical information. The weather API provides enough weather data for basic weather information (eg current weather, forecast, UV index data, and historical weather information). You can use geolocation and names to get a city location.

Chapter 2
Problem Identification

2.1 Problem Statement

Weather prediction is a useful tool for informing populations of expected weather conditions. Weather prediction is a complex topic and poses significant variation in practice. We will attempt to understand and implement a weather prediction application using the linear regression.

2.2 Project Scope

- Weather forecasts are made by collecting as much data as possible about the current state of the atmosphere (particularly the temperature, humidity and wind) to determine how the atmosphere evolves in the future.
- However, the chaotic nature of the atmosphere makes the forecasts less accurate as the range of the forecast increases.
- Traditional observations made at the surface of atmospheric pressure, temperature, wind speed, wind direction, humidity, precipitation are collected routinely from trained observers, automatic weather stations or buoys.During the data assimilation process, information gained from the observations is used In conjunction with a numerical model's most recent forecast for the time that observations were made to produce the meteorological analysis. The complicated equations which govern how the state of a fluid changes with time require supercomputers to solve them.
- The output from this model can be used the weather forecast as alternative.

2.3 Design and Implementation Constraints

The product is developed using API server. The back-end database are CSV files on the basis of that, the prediction takes place. The product is a general-purpose application software in which any user can gather the predicted information. The linear regression is used that predicts the weather based on the previous analysis of a data.
Major Components:

- Data Collection: The feeding historical data to the system, this could be from specific region.
- Data Cleaning: Under this component the data like the missing data, duplicated data is found and bad data is weed out
- Data selection: Under this stage, relevant data related to analysis is retrieved and classified under 6 attributes.

2.4 User Documentation

The product is a general-purpose application software where any user can access the software to gather information about the weather based on a current or a previous data analysis. In this product the software makes a request to the server to access the current dataset through which the data is analyzed using various machine learning algorithms.

2.5 Assumptions and Dependencies

- The software product is dependent on a dataset that is been retrieved from the server using various commands.
- The product will work based on the algorithm that has been discussed above.

2.6 System Features

2.6.1 Actors

- User
- Historical data provider
- Administrator

2.7 Functional Requirements

2.7.1 Accessing a database

- The system should allow administrator to add historical weather data.
- The system should be able to recognize patterns in temperature, humidity, and wind with use of historical data.

2.7.2 Prediction algorithm

- System should periodically apply prediction algorithms or models on obtained data and store results to central database.
- System shall obtain and display confidence value for each prediction given to user.

2.7.3 Actions performed by system

- System shall allow users to check weather for future three days.

2.8 Non-Functional Requirements

2.8.1 User Non-Functional Requirements

- System shall allow for users to get prediction for weather within almost two mouse clicks.
- System should ensure that features that do not require a user to be logged in.

2.8.2 System Non-Functional Requirements

- System should be able to run with core functionality from computer system.

- System should be able to show interactive animations to users regarding current and future climatic conditions.

2.8.3 Other Non-Functional Requirements

- System should textual prediction of climate conditions.

2.9 Other Requirements

2.9.1 Performance Requirements

The proposed software that we are going to develop will be used as the general-purpose application software. Therefore, it is expected that the database would perform functionally all the requirements that are specified by the user.

2.9.2 Safety Requirements

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.

2.9.3 Security Requirements

We are going to develop a secured database for the user. Software Quality Attributes. The Quality of the database is maintained in such a way so that it can be very user friendly to all the users.

2.9.4 Hardware Requirements

The system requires a database in order to store persistent data.

2.9.5 Software Constraints

The development of the system will be constrained by the availability of required software such as web servers, datasetand development tools.

Chapter 3
Testing

Testing is the process of evaluating a system or its component with the intent to find whether it satisfies the specified requirement or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. Systems should not be tested as a single, monolithic unit. The testing process should therefore proceed in the stages where testing is carried out incrementally in conjunction with system implementation. Errors in program components may come to light at a later stage of the testing process. The process is therefore an iterative one with information being fed back from later stage to earlier parts of the process. Following testings were done during the course of our project.

3.1 Unit Testing

Unit testing focuses verification efforts on the smaller unit of software design. Using the detailed design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of the test and the error detected as a result is limited by the constraint scope established for unit testing. The unit test is always white box oriented, and the step can be conducted in parallel for multiple modules

- Tested individual python file by debugging and using print statement
- Individual Component rendering

3.2 Integration Testing

With unit testing the modules may function properly, but at times they may have inadvertent affect on another, sub function when combined, may not produce the desired functions; individually acceptable impression may be signed to unacceptable levels then global data structure may present problems. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by the design.

Chapter 4

Deployment

4.1 Purpose of Deployment Phase

The deployment phase is the final phase of the software development life cycle (SDLC) and puts the product into production. After the project team tests the product and the product passes each testing phase, the product is ready to go live. This means that the product is ready to be used in a real environment by all end users of the product.

There are various phases of the deployment process the project team must follow to ensure the code and technology deploy appropriately. The phases include deployment preparation and procedures, product deployment, transferring ownership of the product, and closing the deployment phase.

4.2 Preparation and Procedures

In the preparation and procedures phase, the project team installs the software and conducts another test to ensure successful installation. Once the installation is complete, the project team creates operating procedures, which include instructions for how the software should work in the information technology environment. If there are issues with system functionality, the operating instructions also provide a mitigation plan to help the end user repair the issue.

4.3 Product Deployment

Under the product deployment phase, the project team implements the programming and coding to each system location. For example, say a company has two regional worksites in Noida and Kolkata and over 2,000 computer systems. The deployment phase includes pushing the program and coding to each regional site and each computer system.

4.4 Environment Variables:

- **Visual Studio Code** - Visual Studio Code is an IDE developed by Microsoft for Windows, Linux and macOS It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent completion, snippets, and code refactoring. It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open source and released under the permissive MIT License. The compiled binaries are freeware and free for private or commercial use.

Working with Git and other SCM providers has never been easier. Review diffs, stage files, and make commits right from the editor. Push and pull from any hosted SCM service. If want to add new languages, themes, debuggers, and to connect to additional services just install the extensions. Extensions run in separate processes, ensuring they won't slow down the editor.

- **Adobe Illustrator**- Adobe Illustrator is a software application for creating drawings, illustrations, and artwork using a Windows or MacOS computer. Illustrator was initially released in 1987 and it continues to be updated at regular intervals, and is now included as part of the Adobe Creative Cloud. Illustrator is widely used by graphic designers, web designers, visual artists, and professional illustrators throughout the world to create high quality artwork. Illustrator includes many sophisticated drawing tools that can reduce the time need to create illustrations

  One of Adobe Illustrator's most important features is that the quality of artwork created using Illustrator is independent of the resolution at which it is displayed. This means that an image created in Illustrator can be enlarged or reduced without sacrificing image quality. This is an attribute of vector artwork, which uses mathematical relationships in describing lines, arcs, and other parts of an illustrator. By comparison, photographs edited using tools such as Adobe Photoshop are resolution-dependent, and image quality decreases when an image is enlarged.

CONCLUSION

"Weather Forecasting Using Time Series Analysis" is a project that has significantly contributed to the field of meteorology and its applications. It is poised to continue providing accurate and timely weather forecasts, supporting decision-making across various sectors, and ultimately improving the quality of life for individuals and communities. The project's findings and methodologies hold the promise of revolutionizing how we approach weather forecasting, making it an invaluable asset for both professionals and the general public.

In conclusion, the project "Weather Forecasting Using Time Series Analysis" has successfully addressed the critical need for accurate and reliable weather forecasting. By leveraging time series analysis techniques and advanced modeling approaches, we have made significant progress in enhancing the accuracy of weather predictions. This project has contributed valuable insights and methodologies to the field of meteorology, benefiting numerous sectors that rely on weather forecasts for decision-making and daily operations.

Through rigorous data collection, preprocessing, and model selection, we have established a robust foundation for forecasting models. The application of various time series analysis methods, including ARIMA, seasonal decomposition, and machine learning algorithms, has enabled us to develop forecasting models that exhibit promising predictive performance. These models have undergone thorough evaluation, with metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) demonstrating the improved accuracy of our forecasts. The project has also emphasized the importance of data visualization to effectively communicate forecast results and model diagnostics. Visualizations have played a crucial role in conveying the forecasted weather conditions to a diverse range of users, from farmers and transportation professionals to emergency response teams and the general public.

FUTURE SCOPE

Scope of Weather Prediction

- Our system will only provide weather prediction of **Jena Climate** only.
- Prediction will be done based on historical weather activities like based on past temperature, wind, etc. pattern what will be the future weather.

Future Enhancement

- Mobile and IOS application Integration.
- Addition of new cities weather dataset to predict there future weather also.
- Addition of new Indices.
- Animation like snow and functions like notifications can also be added.

REFERENCES

1. https.wikipedia.com
2. https.w3schools.com
3. https.reactjs.org
- https://storage.googleapis.com/tensorflow/tf-keras-datasets/jena_climate_2009_2016.csv.zip

- **CODE AND OUTPUTS:**

[38]:
```
# time series library import
import os
import datetime
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import plotly.express as px
from statsmodels.tsa.seasonal import seasonal_decompose
```

[4]:
```
# managing figure size in the begining of the code
mpl.rcParams['figure.figsize'] = (15,6)
mpl.rcParams['axes.grid'] = False
```

[5]:
```
#import the dataset
zip_path = tf.keras.utils.get_file(
    origin = 'https://storage.googleapis.com/tensorflow/tf-keras-datasets/
↪jena_climate_2009_2016.csv.zip',
    fname = 'jena_climate_2009_2016.csv.zip',
    extract = True)
csv_path,_ = os.path.splitext(zip_path)
```

[6]:
```
# read the data in pandas dataframedf
= pd.read_csv(csv_path) df.head()
```

[6]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)
\0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1

	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	\
0	3.33	3.11	0.22	1.94	3.12	

1	3.23	3.02	0.21	1.89	3.03
2	3.21	3.01	0.20	1.88	3.02
3	3.26	3.07	0.19	1.92	3.08
4	3.27	3.08	0.19	1.92	3.09

	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	1307.75	1.03	1.75	152.3
1	1309.80	0.72	1.50	136.1
2	1310.24	0.19	0.63	171.6
3	1309.19	0.34	0.50	198.0
4	1309.00	0.32	0.63	214.3

[7]:
```
# read the data in pandas dataframe
df = pd.read_csv(csv_path)
# write the csv file
df.to_csv('jena_climate_2009_2016.csv')
```

[8]:
```
df.head()
```

[8]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	\
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	

	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	\
0	3.33	3.11	0.22	1.94	3.12	
1	3.23	3.02	0.21	1.89	3.03	
2	3.21	3.01	0.20	1.88	3.02	
3	3.26	3.07	0.19	1.92	3.08	
4	3.27	3.08	0.19	1.92	3.09	

	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	1307.75	1.03	1.75	152.3
1	1309.80	0.72	1.50	136.1
2	1310.24	0.19	0.63	171.6
3	1309.19	0.34	0.50	198.0
4	1309.00	0.32	0.63	214.3

[9]:
```
df.shape
```

[9]: (420551, 15)

[10]:
```
420551*15
```

[10]: 6308265

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 420551 entries, 0 to 420550
Data columns (total 15 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   Date Time        420551 non-null  object 
 1   p (mbar)         420551 non-null  float64
 2   T (degC)         420551 non-null  float64
 3   Tpot (K)         420551 non-null  float64
 4   Tdew (degC)      420551 non-null  float64
 5   rh (%)           420551 non-null  float64
 6   VPmax (mbar)     420551 non-null  float64
 7   VPact (mbar)     420551 non-null  float64
 8   VPdef (mbar)     420551 non-null  float64
 9   sh (g/kg)        420551 non-null  float64
 10  H2OC (mmol/mol)  420551 non-null  float64
 11  rho (g/m**3)     420551 non-null  float64
 12  wv (m/s)         420551 non-null  float64
 13  max. wv (m/s)    420551 non-null  float64
 14  wd (deg)         420551 non-null  float64
dtypes: float64(14), object(1)
memory usage: 48.1+ MB
```

[12]:
```
df.describe()
```

[12]:

	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	\
count	420551.000000	420551.000000	420551.000000	420551.000000	
mean	989.212776	9.450147	283.492743	4.955854	
std	8.358481	8.423365	8.504471	6.730674	
min	913.600000	-23.010000	250.600000	-25.010000	
25%	984.200000	3.360000	277.430000	0.240000	
50%	989.580000	9.420000	283.470000	5.220000	
75%	994.720000	15.470000	289.530000	10.070000	
max	1015.350000	37.280000	311.340000	23.110000	

	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	\
count	420551.000000	420551.000000	420551.000000	420551.000000	
mean	76.008259	13.576251	9.533756	4.042412	
std	16.476175	7.739020	4.184164	4.896851	
min	12.950000	0.950000	0.790000	0.000000	
25%	65.210000	7.780000	6.210000	0.870000	
50%	79.300000	11.820000	8.860000	2.190000	
75%	89.400000	17.600000	12.350000	5.300000	
max	100.000000	63.770000	28.320000	46.010000	

	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	\
count	420551.000000	420551.000000	420551.000000	420551.000000	
mean	6.022408	9.640223	1216.062748	1.702224	
std	2.656139	4.235395	39.975208	65.446714	
min	0.500000	0.800000	1059.450000	-9999.000000	
25%	3.920000	6.290000	1187.490000	0.990000	
50%	5.590000	8.960000	1213.790000	1.760000	
75%	7.800000	12.490000	1242.770000	2.860000	
max	18.130000	28.820000	1393.540000	28.490000	

	max. wv (m/s)	wd (deg)
count	420551.000000	420551.000000
mean	3.056555	174.743738
std	69.016932	86.681693
min	-9999.000000	0.000000
25%	1.760000	124.900000
50%	2.960000	198.100000
75%	4.740000	234.100000
max	23.500000	360.000000

[13]:
```
df.describe().transpose()
```

[13]:

	count	mean	std	min	25%	50%	\
p (mbar)	420551.0	989.212776	8.358481	913.60	984.20	989.58	
T (degC)	420551.0	9.450147	8.423365	-23.01	3.36	9.42	
Tpot (K)	420551.0	283.492743	8.504471	250.60	277.43	283.47	
Tdew (degC)	420551.0	4.955854	6.730674	-25.01	0.24	5.22	
rh (%)	420551.0	76.008259	16.476175	12.95	65.21	79.30	
VPmax (mbar)	420551.0	13.576251	7.739020	0.95	7.78	11.82	
VPact (mbar)	420551.0	9.533756	4.184164	0.79	6.21	8.86	
VPdef (mbar)	420551.0	4.042412	4.896851	0.00	0.87	2.19	
sh (g/kg)	420551.0	6.022408	2.656139	0.50	3.92	5.59	
H2OC (mmol/mol)	420551.0	9.640223	4.235395	0.80	6.29	8.96	
rho (g/m**3)	420551.0	1216.062748	39.975208	1059.45	1187.49	1213.79	
wv (m/s)	420551.0	1.702224	65.446714	-9999.00	0.99	1.76	
max. wv (m/s)	420551.0	3.056555	69.016932	-9999.00	1.76	2.96	
wd (deg)	420551.0	174.743738	86.681693	0.00	124.90	198.10	

	75%	max
p (mbar)	994.72	1015.35
T (degC)	15.47	37.28
Tpot (K)	289.53	311.34
Tdew (degC)	10.07	23.11
rh (%)	89.40	100.00
VPmax (mbar)	17.60	63.77
VPact (mbar)	12.35	28.32
VPdef (mbar)	5.30	46.01

sh (g/kg)	7.80	18.13
H2OC (mmol/mol)	12.49	28.82
rho (g/m**3)	1242.77	1393.54
wv (m/s)	2.86	28.49
max. wv (m/s)	4.74	23.50
wd (deg)	234.10	360.00

[14]:
```
df.head(100)
```

[14]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	\
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	
..	...	...	...	...	...	...	
95	01.01.2009 16:00:00	999.94	-5.40	267.76	-6.86	89.4	
96	01.01.2009 16:10:00	1000.05	-5.31	267.85	-6.69	89.9	
97	01.01.2009 16:20:00	1000.05	-5.28	267.88	-6.68	89.8	
98	01.01.2009 16:30:00	1000.10	-5.32	267.83	-6.77	89.5	
99	01.01.2009 16:40:00	1000.17	-5.29	267.86	-6.70	89.7	

	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	\
0	3.33	3.11	0.22	1.94	3.12	
1	3.23	3.02	0.21	1.89	3.03	
2	3.21	3.01	0.20	1.88	3.02	
3	3.26	3.07	0.19	1.92	3.08	
4	3.27	3.08	0.19	1.92	3.09	
..	...	...	...	...	...	
95	4.08	3.65	0.43	2.27	3.65	
96	4.11	3.69	0.42	2.30	3.69	
97	4.12	3.70	0.42	2.30	3.70	
98	4.11	3.67	0.43	2.29	3.67	
99	4.12	3.69	0.42	2.30	3.69	

	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	1307.75	1.03	1.75	152.3
1	1309.80	0.72	1.50	136.1
2	1310.24	0.19	0.63	171.6
3	1309.19	0.34	0.50	198.0
4	1309.00	0.32	0.63	214.3
..	...	...	...	...
95	1299.17	1.40	2.13	145.5
96	1298.81	1.03	1.50	127.0
97	1298.69	1.11	1.88	134.7
98	1298.96	1.35	2.13	132.3
99	1298.87	1.03	1.75	140.8

[100 rows x 15 columns]

[15]:
```
df.tail(100)
```

[15]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	\
420451	31.12.2016 07:30:00	1006.34	-6.26	266.42	-9.32	
420452	31.12.2016 07:40:00	1006.26	-6.35	266.33	-8.62	
420453	31.12.2016 07:50:00	1006.29	-6.46	266.22	-8.80	
420454	31.12.2016 08:00:00	1006.28	-6.80	265.89	-8.85	
420455	31.12.2016 08:10:00	1006.22	-6.84	265.85	-8.71	
...	...	...	...	...	...	
420546	31.12.2016 23:20:00	1000.07	-4.05	269.10	-8.13	
420547	31.12.2016 23:30:00	999.93	-3.35	269.81	-8.06	
420548	31.12.2016 23:40:00	999.82	-3.16	270.01	-8.21	
420549	31.12.2016 23:50:00	999.81	-4.23	268.94	-8.53	
420550	01.01.2017 00:00:00	999.82	-4.82	268.36	-8.42	

	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	\
420451	78.70	3.82	3.01	0.81	1.86	
420452	83.80	3.79	3.18	0.61	1.97	
420453	83.30	3.76	3.13	0.63	1.94	
420454	85.20	3.67	3.12	0.54	1.93	
420455	86.40	3.65	3.16	0.50	1.95	
...	...	...	...	...	...	
420546	73.10	4.52	3.30	1.22	2.06	
420547	69.71	4.77	3.32	1.44	2.07	
420548	67.91	4.84	3.28	1.55	2.05	
420549	71.80	4.46	3.20	1.26	1.99	
420550	75.70	4.27	3.23	1.04	2.01	

	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
420451	2.99	1311.97	1.20	1.68	163.9
420452	3.16	1312.27	0.60	1.16	145.7
420453	3.11	1312.85	0.50	1.04	184.1
420454	3.10	1314.51	0.74	1.76	178.1
420455	3.14	1314.62	0.95	1.90	181.1
...	...	...	...	...	...
420546	3.30	1292.98	0.67	1.52	240.0
420547	3.32	1289.44	1.14	1.92	234.3
420548	3.28	1288.39	1.08	2.00	215.2
420549	3.20	1293.56	1.49	2.16	225.8
420550	3.23	1296.38	1.23	1.96	184.9

[100 rows x 15 columns]

[16]:
```
# print all the columns
pd.set_option('display.max_rows', None) df.head(20)
```

[16]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	\
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	
5	01.01.2009 01:00:00	996.50	-8.05	265.38	-8.78	94.4	
6	01.01.2009 01:10:00	996.50	-7.62	265.81	-8.30	94.8	
7	01.01.2009 01:20:00	996.50	-7.62	265.81	-8.36	94.4	
8	01.01.2009 01:30:00	996.50	-7.91	265.52	-8.73	93.8	
9	01.01.2009 01:40:00	996.53	-8.43	264.99	-9.34	93.1	
10	01.01.2009 01:50:00	996.62	-8.76	264.66	-9.66	93.1	
11	01.01.2009 02:00:00	996.62	-8.88	264.54	-9.77	93.2	
12	01.01.2009 02:10:00	996.63	-8.85	264.57	-9.70	93.5	
13	01.01.2009 02:20:00	996.74	-8.83	264.58	-9.68	93.5	
14	01.01.2009 02:30:00	996.81	-8.66	264.74	-9.46	93.9	
15	01.01.2009 02:40:00	996.81	-8.66	264.74	-9.50	93.6	
16	01.01.2009 02:50:00	996.86	-8.70	264.70	-9.55	93.5	
17	01.01.2009 03:00:00	996.84	-8.81	264.59	-9.66	93.5	
18	01.01.2009 03:10:00	996.87	-8.84	264.56	-9.69	93.5	
19	01.01.2009 03:20:00	996.97	-8.94	264.45	-9.82	93.3	

	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	\
0	3.33	3.11	0.22	1.94	3.12	
1	3.23	3.02	0.21	1.89	3.03	
2	3.21	3.01	0.20	1.88	3.02	
3	3.26	3.07	0.19	1.92	3.08	
4	3.27	3.08	0.19	1.92	3.09	
5	3.33	3.14	0.19	1.96	3.15	
6	3.44	3.26	0.18	2.04	3.27	
7	3.44	3.25	0.19	2.03	3.26	
8	3.36	3.15	0.21	1.97	3.16	
9	3.23	3.00	0.22	1.88	3.02	
10	3.14	2.93	0.22	1.83	2.94	
11	3.12	2.90	0.21	1.81	2.91	
12	3.12	2.92	0.20	1.82	2.93	
13	3.13	2.92	0.20	1.83	2.93	
14	3.17	2.98	0.19	1.86	2.99	
15	3.17	2.97	0.20	1.85	2.98	
16	3.16	2.95	0.21	1.85	2.96	
17	3.13	2.93	0.20	1.83	2.94	
18	3.13	2.92	0.20	1.83	2.93	
19	3.10	2.89	0.21	1.81	2.90	

	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	1307.75	1.03	1.75	152.3
1	1309.80	0.72	1.50	136.1
2	1310.24	0.19	0.63	171.6
3	1309.19	0.34	0.50	198.0
4	1309.00	0.32	0.63	214.3
5	1307.86	0.21	0.63	192.7
6	1305.68	0.18	0.63	166.5
7	1305.69	0.19	0.50	118.6
8	1307.17	0.28	0.75	188.5
9	1309.85	0.59	0.88	185.0
10	1311.64	0.45	0.88	183.2
11	1312.25	0.25	0.63	190.3
12	1312.11	0.16	0.50	158.3
13	1312.15	0.36	0.63	184.8
14	1311.37	0.33	0.75	155.9
15	1311.38	0.07	0.50	272.4
16	1311.64	0.32	0.63	219.2
17	1312.18	0.18	0.63	167.2
18	1312.37	0.07	0.25	129.3
19	1313.01	0.10	0.63	115.3

[17]:
```
# select the data based on the Date Time column for every 30 minutes interval
# Slice [start: stop: step], starting from index 11 take every 12th record.
df= df[11::12]
df.head(10)
```

[17]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	\
11	01.01.2009 02:00:00	996.62	-8.88	264.54	-9.77	93.2	
23	01.01.2009 04:00:00	996.99	-9.05	264.34	-10.02	92.6	
35	01.01.2009 06:00:00	997.71	-9.67	263.66	-10.62	92.7	
47	01.01.2009 08:00:00	999.17	-8.10	265.12	-9.05	92.8	
59	01.01.2009 10:00:00	1000.27	-7.04	266.10	-8.17	91.6	
71	01.01.2009 12:00:00	1000.30	-6.87	266.27	-8.28	89.6	
83	01.01.2009 14:00:00	999.81	-5.94	267.24	-7.43	89.1	
95	01.01.2009 16:00:00	999.94	-5.40	267.76	-6.86	89.4	
107	01.01.2009 18:00:00	1000.16	-5.25	267.90	-6.75	89.1	
119	01.01.2009 20:00:00	1000.22	-4.90	268.24	-6.38	89.3	

	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	\
11	3.12	2.90	0.21	1.81	2.91	
23	3.07	2.85	0.23	1.78	2.85	
35	2.93	2.71	0.21	1.69	2.72	
47	3.31	3.07	0.24	1.92	3.08	
59	3.60	3.30	0.30	2.05	3.29	
71	3.64	3.27	0.38	2.03	3.26	

83	3.92	3.49	0.43	2.17	3.49
95	4.08	3.65	0.43	2.27	3.65
107	4.13	3.68	0.45	2.29	3.68
119	4.24	3.79	0.45	2.36	3.78

	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
11	1312.25	0.25	0.63	190.3
23	1313.61	0.10	0.38	240.0
35	1317.71	0.05	0.50	146.0
47	1311.65	0.72	1.25	213.9
59	1307.76	1.45	3.00	292.6
71	1306.98	1.84	2.63	184.4
83	1301.67	1.25	2.00	144.0
95	1299.17	1.40	2.13	145.5
107	1298.68	0.55	1.00	183.7
119	1297.05	0.68	1.13	195.2

[18]:
```
df.shape
```

[18]: (35045, 15)

[19]:
```
date_time = pd.to_datetime(df.pop('Date Time'), format='%d.%m.%Y %H:%M:%S')
```

[20]:
```
df.head()
```

[20] :

	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	\
11	996.62	-8.88	264.54	-9.77	93.2	3.12	
23	996.99	-9.05	264.34	-10.02	92.6	3.07	
35	997.71	-9.67	263.66	-10.62	92.7	2.93	
47	999.17	-8.10	265.12	-9.05	92.8	3.31	
59	1000.27	-7.04	266.10	-8.17	91.6	3.60	

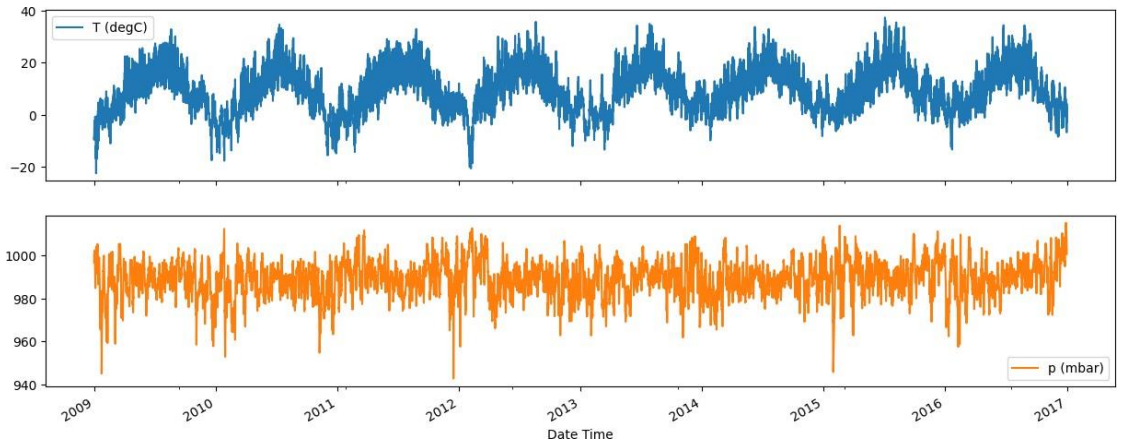
	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	\
11	2.90	0.21	1.81	2.91	1312.25	
23	2.85	0.23	1.78	2.85	1313.61	
35	2.71	0.21	1.69	2.72	1317.71	
47	3.07	0.24	1.92	3.08	1311.65	
59	3.30	0.30	2.05	3.29	1307.76	

	wv (m/s)	max. wv (m/s)	wd (deg)
11	0.25	0.63	190.3
23	0.10	0.38	240.0
35	0.05	0.50	146.0
47	0.72	1.25	213.9
59	1.45	3.00	292.6

[21]:
```
df.columns
```

[21]:
```
Index(['p (mbar)', 'T (degC)', 'Tpot (K)', 'Tdew (degC)', 'rh (%)',
       'VPmax (mbar)', 'VPact (mbar)', 'VPdef (mbar)', 'sh (g/kg)',
       'H2OC (mmol/mol)', 'rho (g/m**3)', 'wv (m/s)', 'max. wv (m/s)',
       'wd (deg)'],
      dtype='object')
```

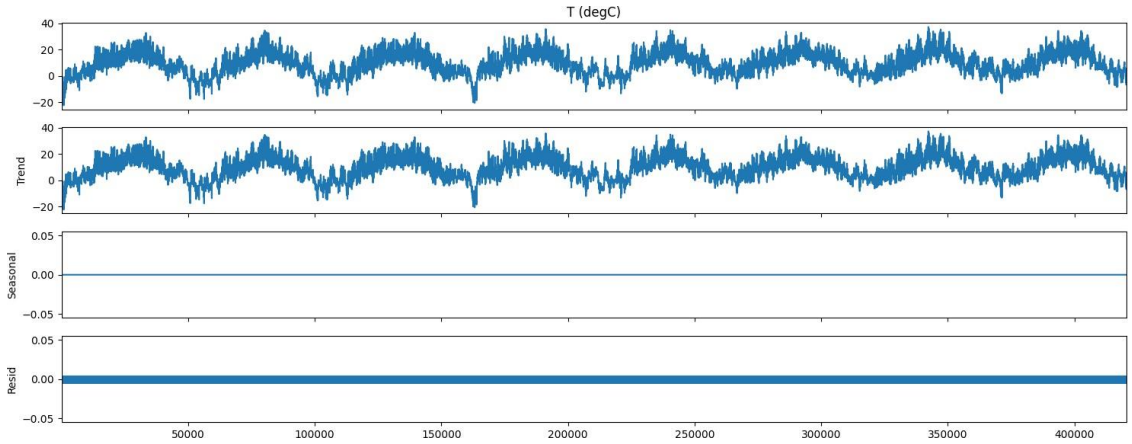
[22]:
```
# let's plot the data
plot_cols = ['T (degC)', 'p (mbar)'] plot_features= df
[plot_cols] plot_features.index = date_time
_ = plot_features.plot(subplots=True)
```

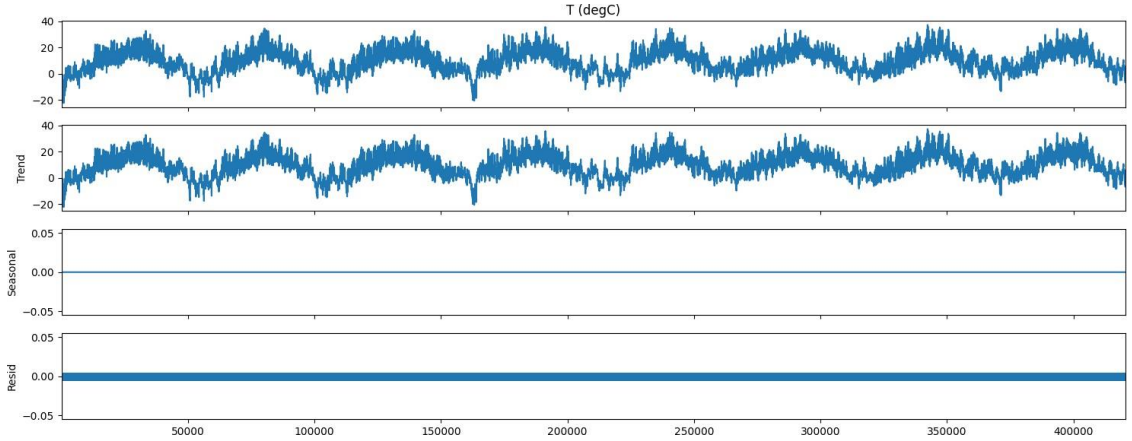


[23]:
```
#lets plot the same plots in plotly
fig = px.line(df, x=date_time, y=plot_cols)
fig.show()
```

[39]:
```
#lets decompose the data into trend, seasonality and noise
# we will use the seasonal decompose function from statsmodels library decompose
= seasonal_decompose (df['T (degC)'], model='additive', period=1) decompose.plot()
```

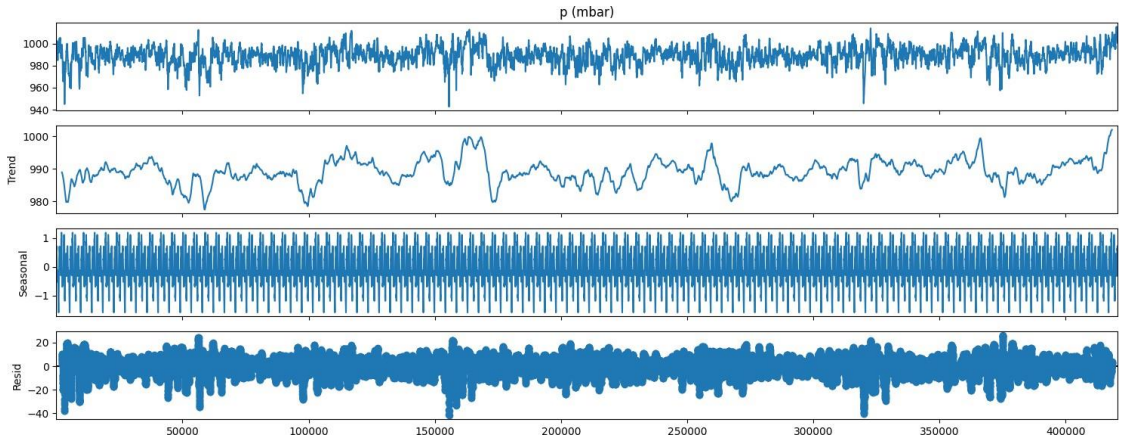
[39]:

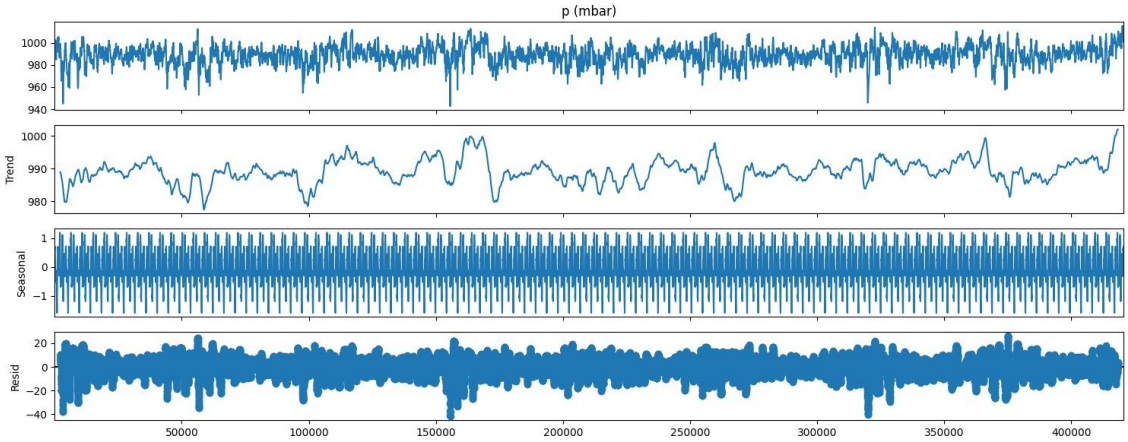




[42]:
```
decompose = seasonal_decompose(df['p (mbar)'], model='additive', period=365)
decompose.plot()
```

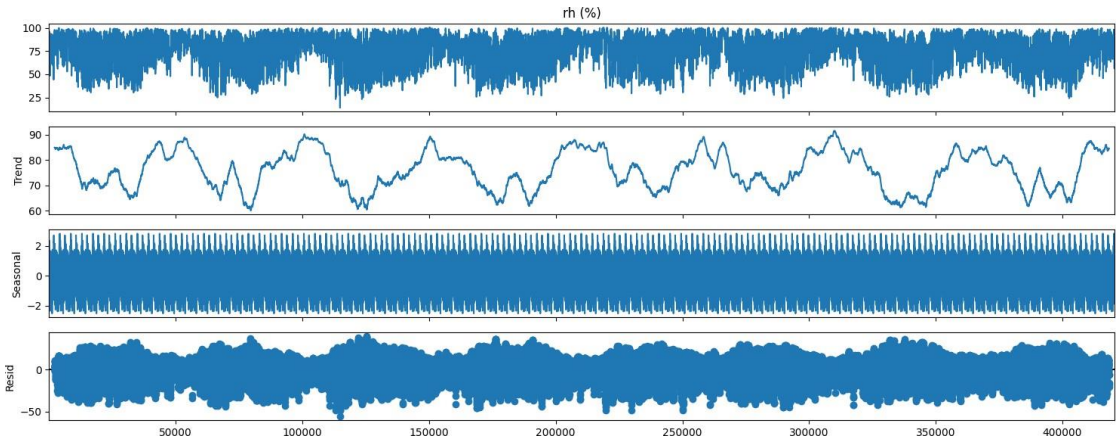
[42]:

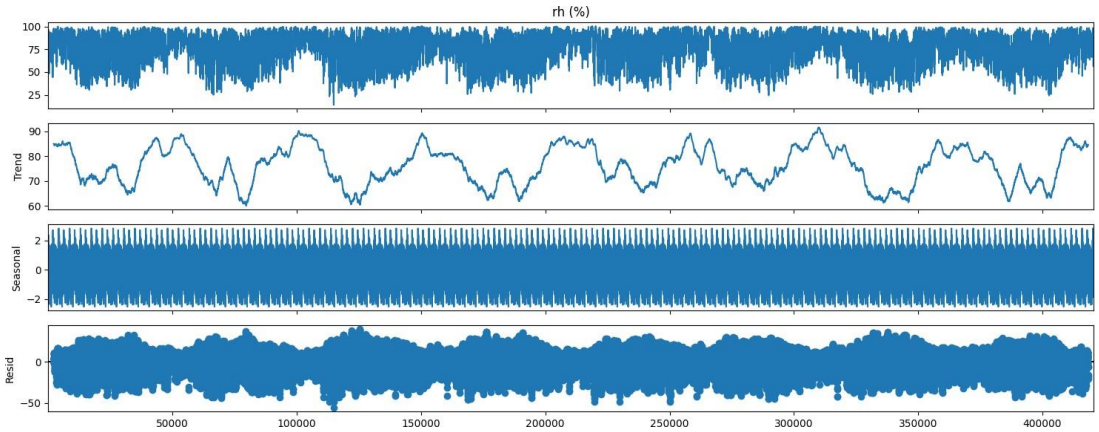




[43]:
```
decompose = seasonal_decompose(df['rh (%)'], model='additive', period=365) decompose.plot()
```

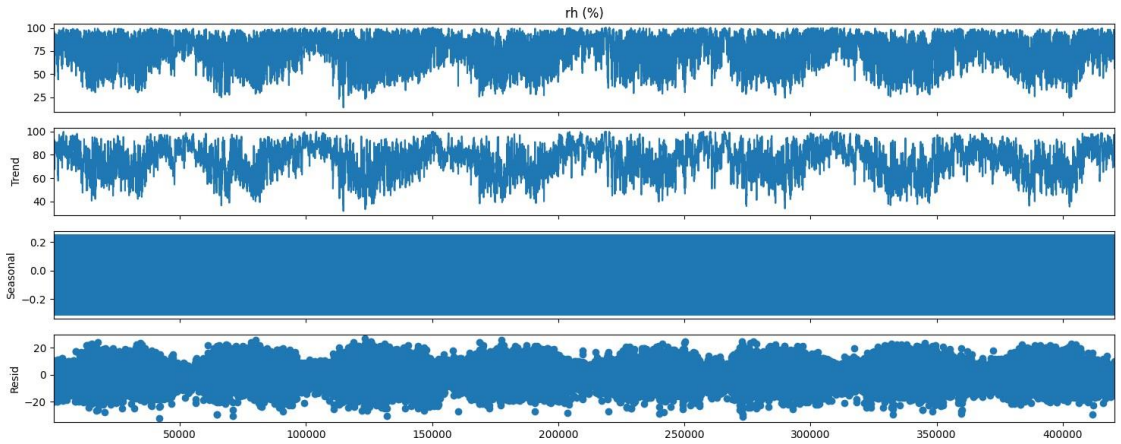
[43]:

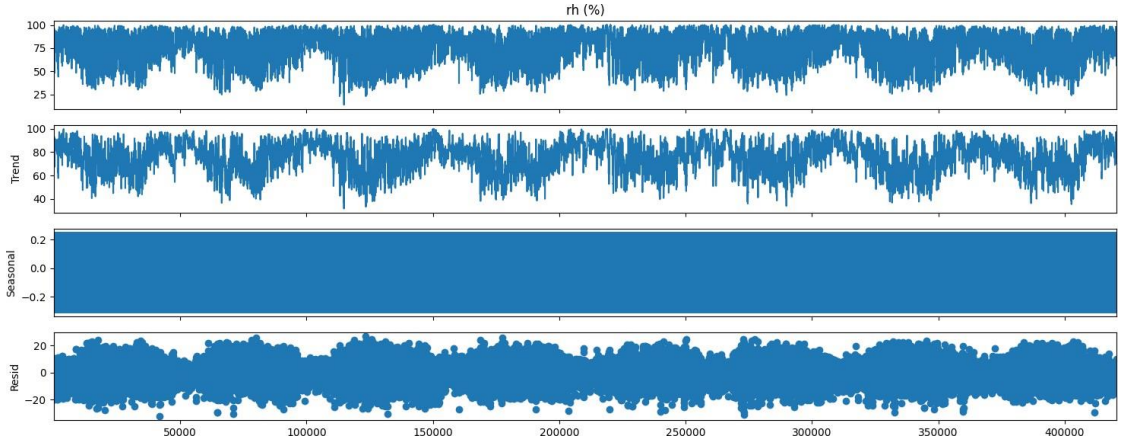




[44]:
```
decompose = seasonal_decompose (df['rh (%)'], model='additive', period=8) decompose.plot()
```

[44]:



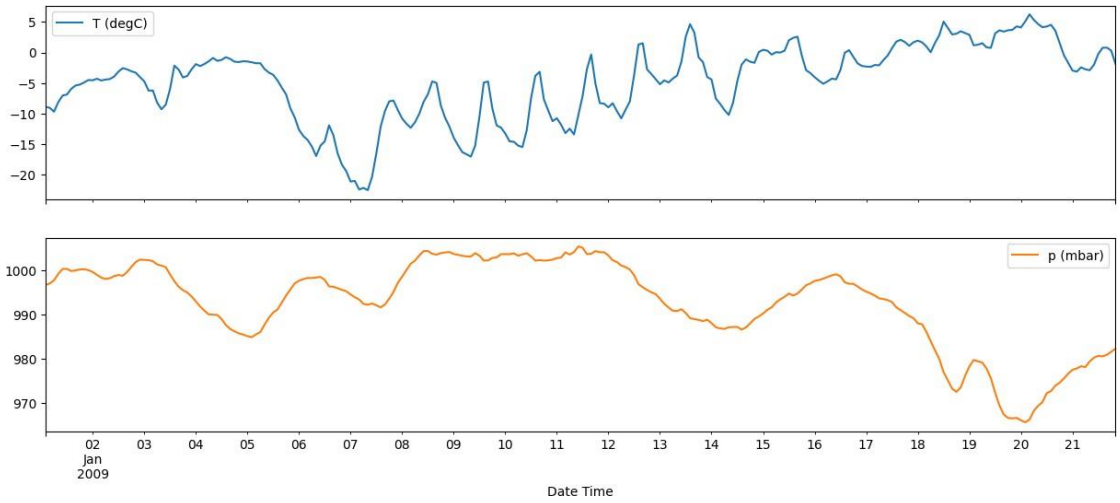


[29]:
```
# lets plot the same plots in plotly
fig = px.line (df, x=date_time, y='T (degC)') fig.show()
fig = px. line (df, x=date_time, y='p (mbar)')
fig.update_traces (line_color='red') fig.show()
```

[32]:
```
# let's plot the data
plot_cols = ['T (degC)', 'p (mbar)']
plot_features = df[plot_cols] [:250]
plot_features.index = date_time [:250]
```

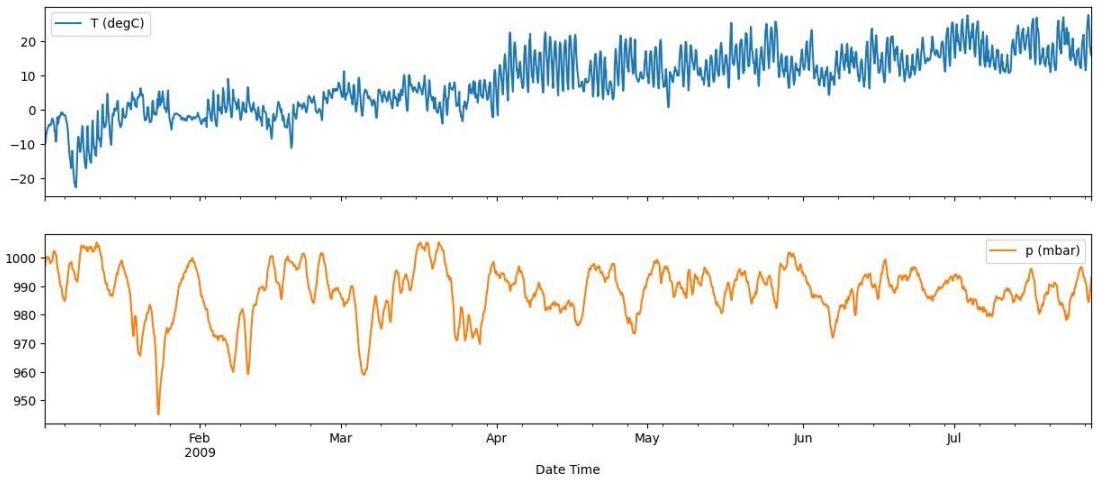
```
plot_features.plot(subplots=True)
```

[32]: array([<Axes: xlabel='Date Time'>, <Axes: xlabel='Date Time'>],
dtype=object)



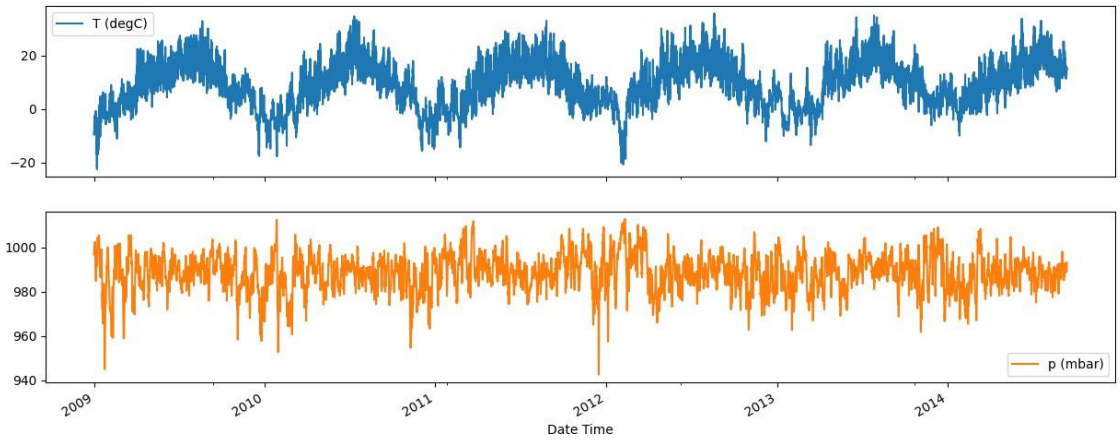
[33]:
```
# let's plot the data
plot_cols = ['T (degC)', 'p (mbar)']
plot_features = df[plot_cols] [:2500]
plot_features.index = date_time [:2500]
plot_features.plot(subplots=True)
```

[33]: array([<Axes: xlabel='Date Time'>, <Axes: xlabel='Date Time'>],
dtype=object)



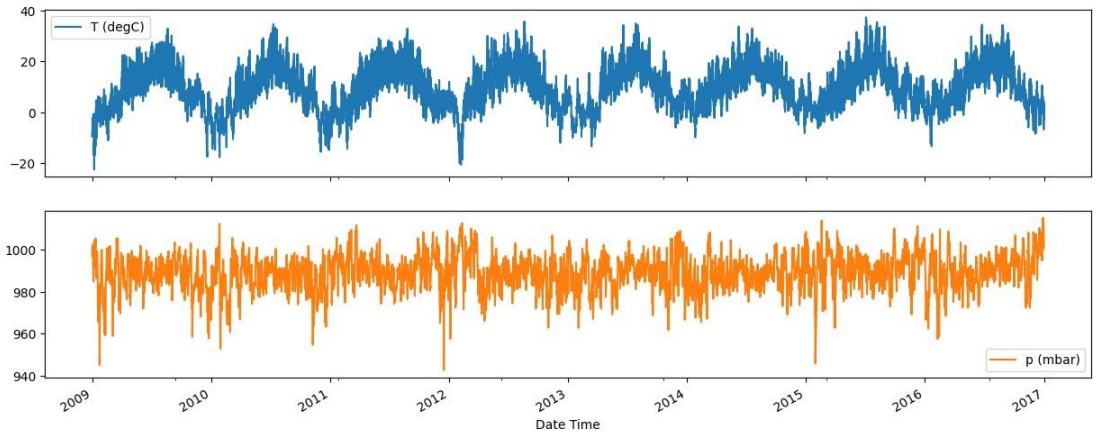
[34]:
```
# let's plot the data
plot_cols = ['T (degC)', 'p (mbar)'] plot_features =
df[plot_cols] [:25000] plot_features.index = date_time
[:25000]plot_features.plot(subplots=True)
```

[34] : array([<Axes: xlabel='Date Time'>, <Axes: xlabel='Date Time'>],
dtype=object)



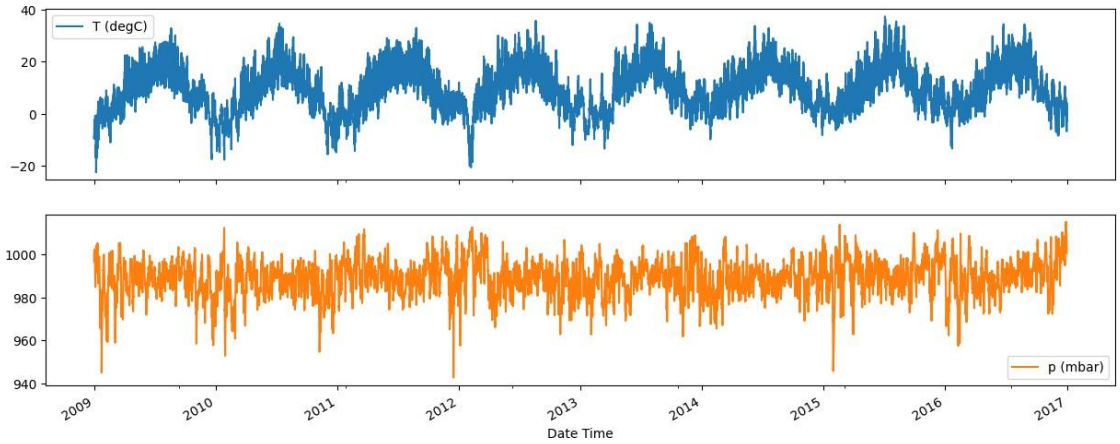
[35]:
```
# let's plot the data
plot_cols = ['T (degC)', 'p (mbar)'] plot_features =
df[plot_cols] [:250000] plot_features.index = date_time
[:250000]plot_features.plot(subplots=True)
```

[35] : array([<Axes: xlabel='Date Time'>, <Axes: xlabel='Date Time'>],
dtype=object)



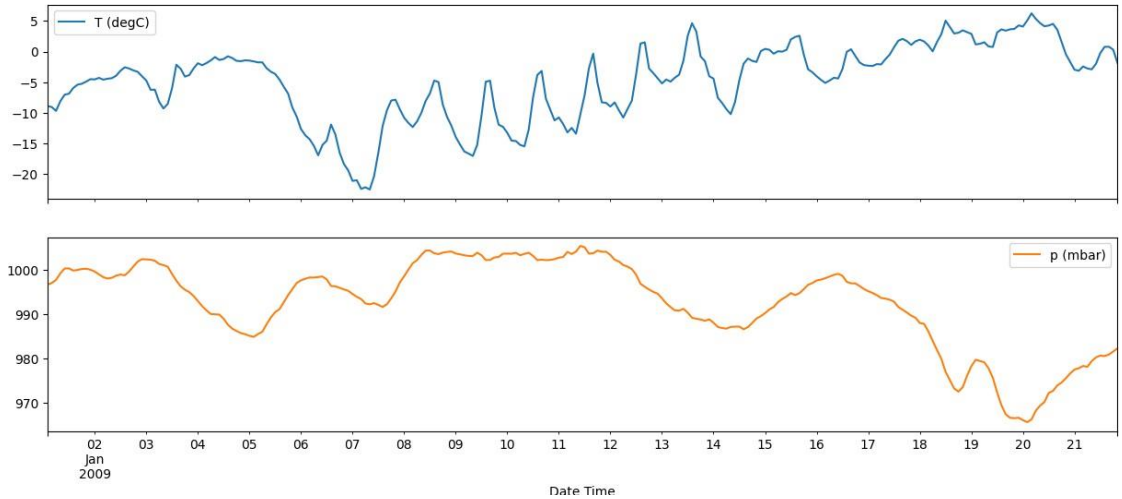
[36]:
```
# let's plot the data
plot_cols = ['T (degC)', 'p (mbar)'] plot_features
= df[plot_cols] [:2500000] plot_features.index =
date_time [:2500000]
plot_features.plot(subplots=True)
```

[36] : array([<Axes: xlabel='Date Time'>, <Axes: xlabel='Date Time'>],
dtype=object)



[37]:
```
# let's plot only the first 500 hours dataset which is equal to 250 records
# let's plot the data
plot_cols = ['T (degC)', 'p (mbar)']
plot_features = df[plot_cols] [:250]
plot_features.index = date_time [:250]
plot_features.plot(subplots=True)
```

[37] : array([<Axes: xlabel='Date Time'>, <Axes: xlabel='Date Time'>],
dtype=object)



[45]:
```
# clean up the data
# lets check if there are any null values in the dataset
df.isnull().sum()
```

[45]:
```
p (mbar)          0
T (degC)          0
Tpot (K)          0
Tdew (degC)       0
rh (%)            0
VPmax (mbar)      0
VPact (mbar)      0
VPdef (mbar)      0
sh (g/kg)         0
H2OC (mmol/mol)   0
rho (g/m**3)      0
wv (m/s)          0
max. wv (m/s)     0
wd (deg)          0
dtype: int64
```

[46]:
```
df.describe().transpose()
```

[46]:

	count	mean	std	min	25%	50%	\
p (mbar)	35045.0	989.213079	8.353850	942.65	984.20	989.57	
T (degC)	35045.0	9.450335	8.425301	-22.50	3.36	9.43	
Tpot (K)	35045.0	283.492908	8.506133	251.22	277.44	283.48	
Tdew (degC)	35045.0	4.955012	6.729168	-24.55	0.24	5.22	
rh (%)	35045.0	76.006020	16.483440	13.88	65.15	79.30	
VPmax (mbar)	35045.0	13.577314	7.743076	0.99	7.77	11.83	
VPact (mbar)	35045.0	9.532714	4.181965	0.83	6.22	8.86	

VPdef (mbar)	35045.0	4.044542	4.902793	0.00	0.87	2.18
sh (g/kg)	35045.0	6.021744	2.654692	0.52	3.92	5.60
H2OC (mmol/mol)	35045.0	9.639148	4.233082	0.83	6.29	8.97
rho (g/m**3)	35045.0	1216.063165	39.978039	1102.46	1187.47	1213.82
wv (m/s)	35045.0	1.558001	75.567534	-9999.00	0.99	1.76
max. wv (m/s)	35045.0	2.961995	75.598740	-9999.00	1.76	2.96
wd (deg)	35045.0	174.660533	86.707469	0.00	125.20	198.10

	75%	max
p (mbar)	994.72	1015.16
T (degC)	15.49	37.28
Tpot (K)	289.55	311.21
Tdew (degC)	10.08	22.19
rh (%)	89.50	100.00
VPmax (mbar)	17.62	63.77
VPact (mbar)	12.36	26.79
VPdef (mbar)	5.30	46.01
sh (g/kg)	7.81	17.13
H2OC (mmol/mol)	12.49	27.25
rho (g/m**3)	1242.80	1391.41
wv (m/s)	2.84	13.50
max. wv (m/s)	4.73	23.50
wd (deg)	233.60	360.00

[49]:
```
wv = df['wv (m/s)'] bad_wv
= WV == -9999.0
wv[bad_wv] = 0.0
```

[53]:
```
max_wv = df ['max. wv (m/s)']
bad_max_wv = max_wv == -9999.0
max_wv [bad_max_wv] = 0.0
```

[54]:
```
df.describe().transpose()
```

[54]:

	count	mean	std	min	25%	50%	\
p (mbar)	35045.0	989.213079	8.353850	942.65	984.20	989.57	
T (degC)	35045.0	9.450335	8.425301	-22.50	3.36	9.43	
Tpot (K)	35045.0	283.492908	8.506133	251.22	277.44	283.48	
Tdew (degC)	35045.0	4.955012	6.729168	-24.55	0.24	5.22	
rh (%)	35045.0	76.006020	16.483440	13.88	65.15	79.30	
VPmax (mbar)	35045.0	13.577314	7.743076	0.99	7.77	11.83	
VPact (mbar)	35045.0	9.532714	4.181965	0.83	6.22	8.86	
VPdef (mbar)	35045.0	4.044542	4.902793	0.00	0.87	2.18	
sh (g/kg)	35045.0	6.021744	2.654692	0.52	3.92	5.60	
H2OC (mmol/mol)	35045.0	9.639148	4.233082	0.83	6.29	8.97	
rho (g/m**3)	35045.0	1216.063165	39.978039	1102.46	1187.47	1213.82	
wv (m/s)	35045.0	2.128639	1.542289	0.00	0.99	1.76	

max. wv (m/s)	35045.0	3.532633	2.343832	0.00	1.76	2.96
wd (deg)	35045.0	174.660533	86.707469	0.00	125.20	198.10

	75%	max
p (mbar)	994.72	1015.16
T (degC)	15.49	37.28
Tpot (K)	289.55	311.21
Tdew (degC)	10.08	22.19
rh (%)	89.50	100.00
VPmax (mbar)	17.62	63.77
VPact (mbar)	12.36	26.79
VPdef (mbar)	5.30	46.01
sh (g/kg)	7.81	17.13
H2OC (mmol/mol)	12.49	27.25
rho (g/m**3)	1242.80	1391.41
wv (m/s)	2.84	13.50
max. wv (m/s)	4.73	23.50
wd (deg)	233.60	360.00

[55]:
```
df.shape
```

[55]: (35045, 14)

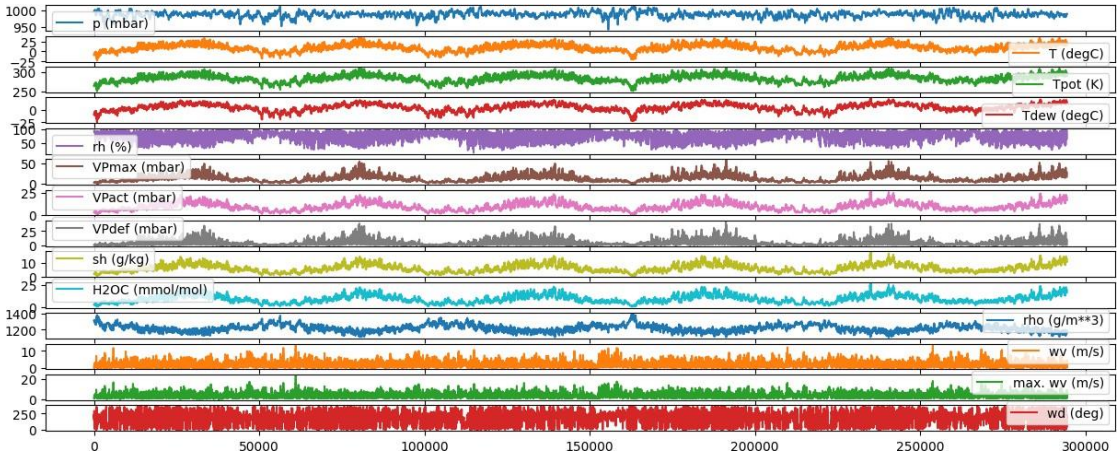
[56]:
```
#split the data into train, validation and test i have 35045 records make 70%, 
  ↪20% for validation and 10% for testing
# 70% of 35045 is 24531
# 20% of 35045 is 7009
# 10% of 35045 is 3505
# total 35045
# train data
train_df = df [:24531]
# validation data
val_df = df [24531:31540]
# test data
test_df = df [31540:]

#lets check the shape of the data
train_df.shape, val_df.shape, test_df.shape
```

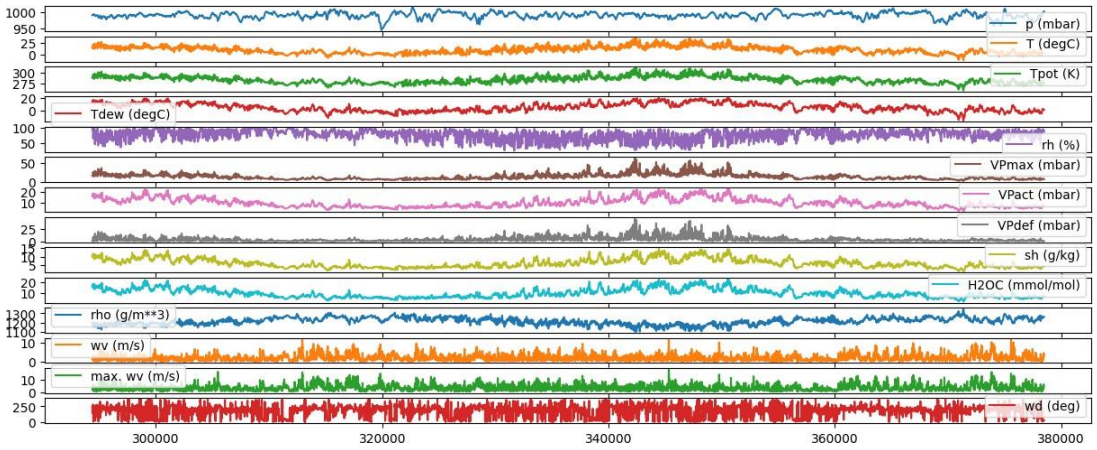
[56]: ((24531, 14), (7009, 14), (3505, 14))

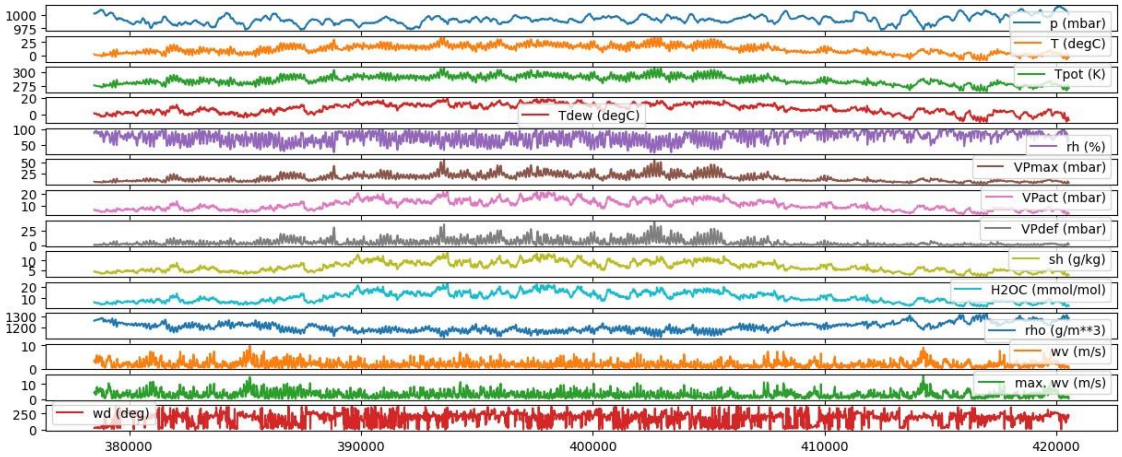
[57]:
```
# lets plot the data
train_df.plot(subplots=True)

# lets plot the data
val_df.plot(subplots=True)
```

```
# lets plot the data
test_df.plot(subplots=True)
```

[57]: array([<Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >,
<Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >,
<Axes: >, <Axes: >], dtype=object)



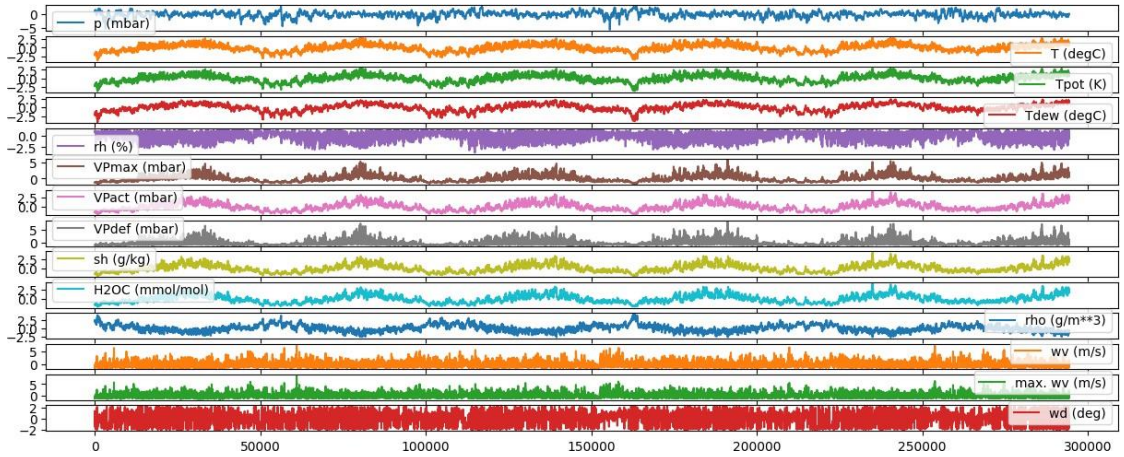


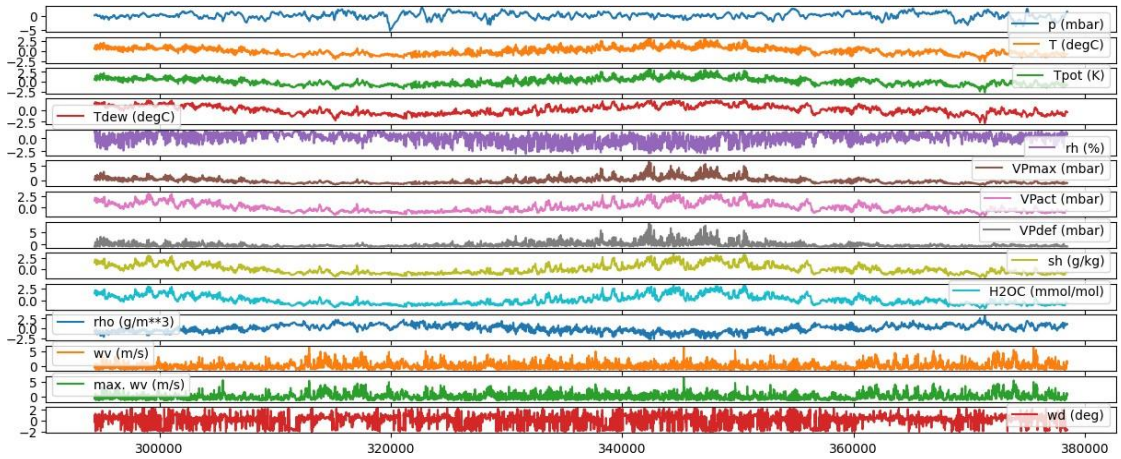


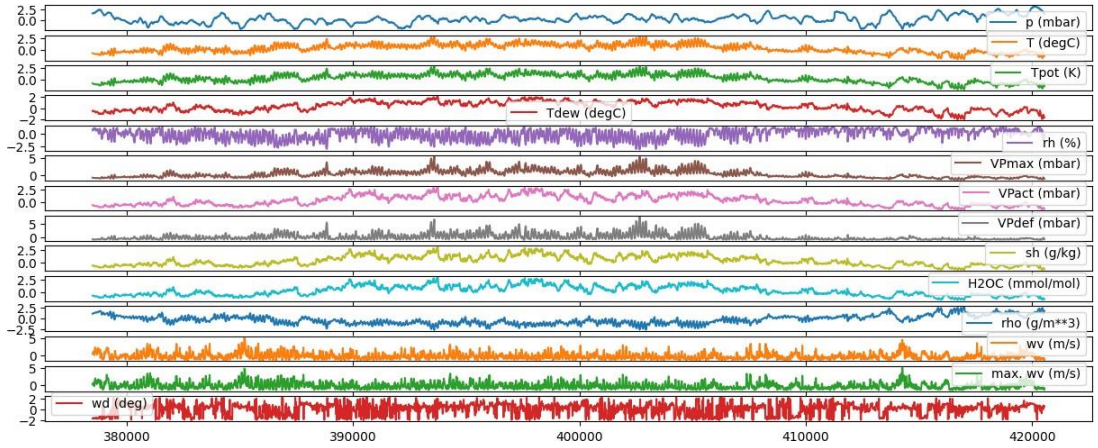
[58]:
```
# lets normalize the data
# we will use the mean and standard deviation of the training data to normalize␣
  ↪the data
train_mean = train_df.mean()
train_std = train_df.std()

#lets normalize the training data
train_df = (train_df - train_mean) / train_std
#lets normalize the validation data
val_df = (val_df - train_mean) / train_std
# lets normalize the test data
```

[59]:
```
# lets plot the data
train_df.plot(subplots=True)

# lets plot the data
val_df.plot(subplots=True)

# lets plot the data
test_df.plot(subplots=True)
```

[59]:
```
array([<Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >,
       <Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >,
       <Axes: >, <Axes: >], dtype=object)
```


p (mbar)
T (degC)
Tpot (K)
Tdew (degC)
rh (%)
VPmax (mbar)
VPact (mbar)
VPdef (mbar)
sh (g/kg)
H2OC (mmol/mol)
rho (g/m**3)
wv (m/s)
max. wv (m/s)
wd (deg)
0
50000
100000
150000
200000
250000
300000


p (mbar)
T (degC)
Tpot (K)
Tdew (degC)
rh (%)
VPmax (mbar)
VPact (mbar)
VPdef (mbar)
sh (g/kg)
H2OC (mmol/mol)
rho (g/m**3)
wv (m/s)
max. wv (m/s)
wd (deg)
300000
320000
340000
360000
380000


p (mbar)
T (degC)
Tpot (K)
Tdew (degC)
rh (%)
VPmax (mbar)
VPact (mbar)
VPdef (mbar)
sh (g/kg)
H2OC (mmol/mol)
rho (g/m**3)
wv (m/s)
max. wv (m/s)
wd (deg)
380000
390000
400000
410000
420000

[ ]:
```
# we will use the windowed dataset to train the model
# we will use the windowed dataset to validate the model
# we will use the windowed dataset to test the model
# we will use the windowed dataset to predict the model
# we will use the windowed dataset to plot the model
```

[ ]:
```
"THANKYOU"
```