Weather Data Analysis

Database Management System Laboratory

Project Report

Submitted by

Abhiroop Sarkar (12022002016041)

Snehal Ghosh (12022002016037)

Under the guidance and supervision of

Prof. Dr. Deepsubhra Guha Roy Submitted to the

Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning and Computer Science and Business Systems)

Institute of Engineering & Management, Kolkata

Acknowledgment

As students from 5th semester of the Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning) at the Institute of Engineering and Management, we submit this project for evaluation for the Database Management Systems Laboratory, PCCCS501(Lab).
We would like to thank our mentor Prof. Dr. Deepshubhra Guha Roy for his support and guidance provided throughout this project.

Contents

1 Introduction

1.1 Problem Statement and Anlysis

The analysis of historical weather data is an important component of climate science, which provides insights into the past atmospheric conditions enabling us for a deeper understanding of climate change and variations. With the rise of global warming and climate change, the need for accurate and comprehensive weather records has never been more pressing. Historical weather data can encompasses a lot lot of variables, like temperature, precipitation, humidity, wind speed, and direction, collected over long periods. This data can serve as a critical resource for researchers to validate climate models, forecast weather, and assess the impacts of human activity on weather patterns.
There has been many advancements in data collection and analysis, which has enhanced our ability to store and utilize weather data effectively. An example of a historical weather database is HCILM [1], which has integrated numerous instrumental records from various sources dating back to the pre-1890 era, providing us with a wealth of information for climate reconstruction. The significance of historical weather data extends beyond the borders of academic research. It plays crucial roles in practical applications across various sectors. For example, agriculture and renewable energy industries rely on this data to make informed decisions about crop management and energy production based on anticipated weather conditions 2. Apart from that, understanding historical weather patterns allows scientists to track changes in climate systems and accurately predict future scenarios. In the early days, fetching current weather data involved people relying on data from nearby weather stations, but technological advancements like API (Application Programming Interfaces) help make weather data more easily available for researchers and industry [2].

1.2 Objective

The primary objectives of our project involve the analysis of past weather data of our city, Kolkata, and presenting it in tabular and pictorial format, which enables the residents and visitors of Kolkata to find a short summary of the climate of the city along with a summary of how the weather has previously been on the present date in the previous year, providing a likely estimate of how the weather of the day is going to turn out that day. We have primarily used MySQL as the database to fetch information regarding the weather, Python with MySQL Connector for the backend operations, and Streamlit module for our frontend design. We have fetched our dataset from Kaggle, which contains weather data of Kolkata [3] from the year 2017 to 2021. It is openly available at https://www.kaggle.com/datasets/kafkarps/five-years-weather-data-of-kolkata.

2 Literature Review

In our literature review, we highlight some of the important models used in weather analysis, including traditional statistical methods, machine learning, and deep learning approaches.

1. **Statistical Methods**

   ARIMA Models

   Autoregressive-integrated-moving-average (ARIMA) and seasonal ARIMA (SARIMA) models are generalized versions of the ARMA model for non-stationary and periodic time-series data. These generalizations so that seasonal variations in data can be observed. ARMA assumes the time series data to be stationary, if there is a constant variance or trend, it is removed through the process of differencing, corresponding to the I (integrated) part of ARIMA, while periodic variation is removed by seasonal differencing, corresponding to the S (seasonal) part of SARIMA. Further generalization was performed which allows external variables that may affect the time series to be included known as Exogenous Regressors, corresponding to X in SARIMAX. [4, 5]

2. **Machine Learning Methods**

   Recent Studies have shown that machine learning models such as Linear Regression, Decision Tree Regressors, and Support Vector Regressors, allow us to utilize weather features such as rain, snow, humidity, and wind to enhance forecasting. Machine learning methods for climate prediction have been evolving since the 1990s. During the early 2000s, the field incorporated machine learning-enhanced models for medium or long-term weather prediction that have been utilized [6]. Examples include Support Vector Machine (SVM)-downscaling and K-Nearest Neighbor (KNN)-downscaling which applied the respective models for regional forecasting. Machine learning models are usually used in bias correction, downscaling, and emulation. They are also used with statistical methods as hybrid models, to refine predictions of statistical and numerical models for accurate predictions.

3. **Deep Learning Methods**

   Deep learning models have significantly advanced climate prediction by offering robust architectures for various temporal and spatial scales. In 1998, early methods like Precipitation Neural Networks used artificial neural networks for short-term forecasting. In 2015, hybrid deep learning models like ConvLSTM combined CNNs and LSTMs for precipitation forecasting [6]. Google's MetNet [7] introduced in 2020, uses neural networks and is tailored specifically for precipitation forecasting, up to 8 hours in the future and was shown to outperform Numerical Weather Prediction at that time interval, by taking radar data, satellite data, and time to forecast ahead as input parameters.

2.1 Types of Weather Forecasting

1. **Long-Term**

   Long-term weather forecasting typically refers to predictions that are made for periods extending beyond two weeks, often ranging from 30 days to several months or even years. This type of forecasting is less focused on specific daily weather events and more on general trends and anomalies over extended periods. This type of forecasting is vital for the agriculture sector, disaster prediction, and energy management, where long-term forecasts can help us understand seasonal trends and help us make strategic decisions.

2. **Short-Term**

   Short-term weather forecasting focuses on predicting atmospheric conditions from a few hours up to a week ahead. This is the type of forecasting we usually see in our weather applications and websites, and at present are quite accurate. Thus, Short-term forecasts are essential for everyday activities, like travel planning, event scheduling, and emergency response during severe weather events. They provide us with critical information that helps individuals and organizations make informed decisions on the go.

3 System Description

3.1 Dataset

Our dataset, sourced from Kaggle [3] consists of multiple weather parameters and location data as features, including:

1. Address: Kolkata, West Bengal, India
2. Date time
3. Minimum Temperature: Daily Minimum
4. Maximum Temperature: Daily Minimum
5. Temperature: Average Temperature
6. Dew Point
7. Relative Humidity
8. Heat Index
9. Wind Speed
10. Wind Gust
11. Wind Direction
12. Wind Chill
13. Precipitation: Daily Rainfall
14. Precipitation Cover
15. Snow Depth
16. Visibility
17. Cloud Cover
18. Sea Level Pressure
19. Weather Type : Like Clear,

20. Latitude
21. Longitude
22. Resolved Address
23. Name
24. Info
25. Conditions

It is presented as a time series of data starting from 1st January 2017, upto 31st December 2022.

During preprocessing we used the *pandas* library of Python to remove the columns Name, Address, Resolved Address, Wind Gust, Wind Chill, Info, Snow Depth, as they were either unnecessary or contained too many NULL values, and therefore better not to include in the database. After that, we converted the dataframe to SQL and pushed it to local MySQL of our system using the *to_sql*() function of pandas and *create_engine*() function of *sqlalchemy* package.

3.2 Database Management System

We use MySQL as the primary database management system in our project. We have written Python Code which is being executed, to query the database to fetch the information that is displayed in the website. Python has been connected to MySQL using the *mysql.connector* available as a module in Python's *mysql* library. MySQL is an open-source relational database management system (RDBMS) which is named after its co-founder Michael Widenius's daughter, "My," combined with "SQL," short for Structured Query Language. It organizes data into related tables, and allows structured storage and easy data retrieval using SQL. MySQL also manages users, enables network access, and supports database integrity and backup testing. It is available as free software under the GNU General Public License, and also offers proprietary licenses [8].

3.3 Back-end

We use Python as the backend for this project. Python libraries such as *mysql.connector*, *pandas*, and *datetime* has been extensively utilized as tools to achieve our purpose.

mysql.connector has been used to connect our Python backend to MySQL database, *pandas* has been used to store and manipulate data in the form of tables, and *datetime* has been used to fetch the current date and time for dynamic features of our web application.

3.4 Front-end

We have used the *Streamlit* framework provided by Python for our front-end. It provides simple, easy-to-use, and clean-looking websites for building webpages using Python only. It also includes support for custom colors using the *toml* file, which we have utilized, and functions to display data in tables. This framework is primarily used for hosting websites for machine learning and data science projects that use Python.

4 Methodology

4.1 File Structure

- .streamlit

  `config.toml`

- `Kolkata_weather_data(2017-2022).csv`
- `Kolkata_weather_data(2017-2022)_processed.csv`
- `Queries.py`
- `main.py`

When the main file is run with streamlit, it first uses the 'config.toml' file in the '.streamlit' folder to configure the parameters of the streamlit site - Parameters such as background colour and font color. It then calls the methods in the Queries file, which executes the queries on the database.

The database itself is seeded with a subset of attributes of the Kolkata weather data file, which contains detailed data of every day from 2014 to 2022. The MySQL database contains the data quite similar to the one in the processed csv.

It contains the following attributes for all the days from 2017 to 2022 for the Kolkata region

Field	Type	Null	Key	Default
Date time	datetime	No	Primary	Null
Minimum Temperature	double	Yes		Null
Maximum Temperature	double	Yes		Null
Temperature	double	Yes		Null
Dew Point	double	Yes		Null
Relative Humidity	double	Yes		Null
Heat Index	double	Yes		Null
Wind Speed	double	Yes		Null
Wind Direction	double	Yes		Null
Precipitation	double	Yes		Null
Precipitation Cover	double	Yes		Null
Visibility	double	Yes		Null
Cloud Cover	double	Yes		Null
Sea Level Pressure	double	Yes		Null
Weather Type	double	Yes		Null
Latitude	double	Yes		Null
Longitude	double	Yes		Null
Conditions	text	Yes		Null

Table 1: Attributes & their datatypes

5 Implementation

```
def dynamic(cursorObject, day, monthNo):
    query=f'''select avg(`Maximum Temperature`), avg(`Minimum
        Temperature`),
            max(`Maximum Temperature`), min(`Minimum Temperature
                `),
            avg(`Precipitation`), max(`Precipitation`)
            from weather where Day(`Date time`)={day} and Month
                (`Date Time`)={monthNo};'''

    cursorObject.execute(query)
    df = pandas.DataFrame(cursorObject.fetchall()[0], columns=['
        '],
                index= ['Average-Max-Temp', 'Average-Min-Temp',
                    'Max-Recorded-Temp', 'Min-Recorded-Temp', '
                    Average-Rainfall', 'Max-Rainfall'])
    return df
```

.

The Dynamic Function helps us to estimate the expected maximum and minimum temperatures for that day of the month across the last 6 years. It also highlights the maximum and minimum recorded temperature for that day of the month. To highlight how wet the day may be, we also give the average and maximum rainfall for that day of month in the last 6 years.

The query is executed and the response is stores into a dataframe with appropriate row labels. The default column label is removed and the dataframe is then returned to the calling function

The function takes the mySQL cursor object pointer along with the day & month numbers as input

```
def conditions_summary(cursorObject, day, monthNo):
    df= pandas.DataFrame(index=[''])

    query = f'''select Conditions, count(Conditions) from
        weather where Day(`Date time`)={day} and Month(`Date Time
        `)={monthNo} group by Conditions'''
    cursorObject.execute(query)
    response = cursorObject.fetchall()

    for condition in ['Clear', 'Partially cloudy', 'Rain', '
        Overcast']:
        sum =0
        for label in response:
            if condition in label[0]:
                sum+= label[1]
        if sum:
            df[condition] = [f"{sum} out of last 6 years "]

    return df
```

.

This function takes the mySQL cursor object pointer along with the day & month numbers as input.

It first initialises and empty dataframe. It then queries the database for the count of each weather condition for that particular day of the month, and fetches the response

It then iterates over a pre-determined list of conditions and searches for it in each of the response items. If a match is found, it adds it to the count for that condition. At last, for each condition, if the count is found to be greater than 0, it is added as a new column in the dataframe.

Finally it returns the dataframe

```
def static_table(cursorObject):
    dataframe = pandas.DataFrame(index= ['Jan', 'Feb', 'Mar', '
        Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', '
        Dec', 'Year'])

    query = '''select min(`Minimum Temperature`) from weather
        group by month(`Date Time`);'''
    cursorObject.execute(query)
    response = cursorObject.fetchall()
    response = [temp[0] for temp in response]
    response.append(min(response))
    dataframe['Min. Recorded Temperature'] =response

    query = '''select avg(`Temperature`) from weather group by
        month(`Date Time`);'''
    cursorObject.execute(query)
    response = cursorObject.fetchall()
    response = [temp[0] for temp in response]
    response.append(sum(response)/len(response))
    dataframe['Average Temperature'] =response

    query = '''select max(`Maximum Temperature`) from weather
        group by month(`Date Time`);'''
    cursorObject.execute(query)
    response = cursorObject.fetchall()
    response = [temp[0] for temp in response]
    response.append(max(response))
    dataframe['Max. Recorded Temperature'] =response

    query = '''select sum(`Precipitation`)/6 from weather group
        by month(`Date Time`);'''
    cursorObject.execute(query)
    response = cursorObject.fetchall()
    response = [temp[0] for temp in response]
    response.append(sum(response))
    dataframe['Average Rainfall'] =response

    query = '''select count(*)/6 from weather where
        Precipitation>0 group by Month(`Date time`);'''
    cursorObject.execute(query)
    response = cursorObject.fetchall()
    response = [float(temp[0]) for temp in response]
    response.append(sum(response))
    dataframe['Average Rainy Days'] =response

    query = '''select avg(`Relative Humidity`) from weather
        group by month(`Date Time`);'''
    cursorObject.execute(query)
    response = cursorObject.fetchall()
    response = [temp[0] for temp in response]
    response.append(sum(response)/len(response))
    dataframe['Average Rel. Humidity'] =response

    return dataframe
```

The static function takes just the mySQL cursorObject as an argument.

It first initialises a DataFrame with the names of the months and year as the index. This dataframe is supposed to be flipped(transposed) before being displayed to the public, making these labels - column headers.

Then it executes the queries which applies an aggregate function on a metric, after grouping them by months. We then run a loop to extract the first object in each tuple of the response. It then applies the aggregate function on this response and append it to the response list. Then the response list is added as a column to the dataframe

At the end, the function returns the dataframe, which is to be transposed before displaying

.

```
import mysql.connector, pandas, toml
import streamlit as st
from datetime import datetime
import Queries as Q
```

.

The first 4 lines of our main python file imports the neccessary libraries.

The 'connector' attribute of the 'mysql' library is used to connect our python files to the mysql service running on the backend server

The 'pandas' library is used for Dataframes

The 'toml' library is neccessary for configuring the streamlit site

The 'streamlit' library is imported as 'st' alias to display our application as a web portal

The 'datetime' attribute of the package of the same name, has the defines the behaviour of dates and time, along with the current date-time and the associated functions

At last the Queries.py file is imported, which has the functions containing our queries

```
def edit_background(file_path, bg, secondary_bg, text):
    with open(file_path, 'r') as file:
        config = toml.load(file)
    config['theme']['backgroundColor'] = bg
    config['theme']['secondaryBackgroundColor'] = secondary_bg
    config['theme']['textColor'] = text
    with open(file_path, 'w') as file:
        toml.dump(config, file)

    print("Background color updated successfully!")

cnx = mysql.connector.connect(user='root', password='12345',
   host='localhost', database='kolkata_weather')
cursor = cnx.cursor()
current_time = datetime.now()
hour = current_time.hour
emoji = ""
if (hour >= 18 or hour < 6):
    edit_background(file_path=".streamlit\config.toml", bg="#102
       b61", secondary_bg="#061024", text="ffffff")
    emoji = ""
else:
    edit_background(file_path=".streamlit\config.toml", bg="#
       e8af1e", secondary_bg="#c9a444", text="#141414")
    emoji = ""
```

We first connect to the server's mySQL database as the root user with the appropriate password and switch to the correct database

We then obtain the cursor object of the connection with the cursor function

We then check the current time. During the daytime (in Kolkata), the site operates in light mode, while it switches to night mode after sunset. The text and background colors of the site are changed accordingly.

```
def main():
    st.set_page_config(layout="wide")
    st.title(f"Weather Data Analysis{emoji}")
    # print(current_time.date)
    navigation_options = ["Home", "Info"]
    menu_selection = st.selectbox("",navigation_options)

    if menu_selection == "Home":
        show_home()
    elif menu_selection == "Info":
        show_info()
```

.

The main function configures the page to be wide by default. It requests to transfer to other pages

.

```
def show_home():
    st.write("### Climate data for Kolkata from 2017 to 2022")
    table = Q.static_table(cursor)
    st.table(data = table.T.style.format("{:.2f}"))
    column1, column2 = st.columns(2)
    with column1:
        st.write(f"### Historic Data for {current_time.date().
            strftime('%d %B')}")
        st.dataframe(Q.conditions_summary(cursor, current_time.
            day, monthNo = current_time.month).T)
    with column2:
        st.dataframe(Q.dynamic(cursor, day=current_time.day,
            monthNo = current_time.month))
```

.

The Home Page calls the Query functions and displays the dataframes. It also has to flip the first dataframe and rounds off the float values to 2 decimal places

.

```
def show_info():
    st.write("# Info")
    st.write("-  ### [Abhiroop's LinkedIn](https://linkedin.com/
        in/abhiroop2004)")
    st.write("-  ### [Snehal's LinkedIn](https://www.linkedin.com
        /in/snehal-ghosh-164a63263)")
    st.write()
    st.write('''Project developed under the guidance of **Prof.
        Dr. Deepshubhra Guha Roy**
              for the **Database Management Systems Laboratory**
              being taught by the department of Computer Science
                 and Engineering
              (Artificial Intelligence and Machine Learning)
                 along with
              Computer Science and Business Studies,
              Institute of Engineering and Management SaltLake'''
                 )
```

.

The Info Page displays the links to the LinkedIn account of the project developers, along with an acknowledgment to the department

6 Conclusion

Climate in Kolkata is one of the basic structuring forces providing the physical conditions and an organizational grid for the existence of the city within which individuals, society and economy are inserted. Located on the banks of Hooghly River, it has a tropical wet and dry climate characteristic of weather condition consisting of hot summer, heavy monsoons, and cool winters. Every season has its opportunities and risks, which intensively influence the socio-economic character of the area.

The hot months are experienced between the months of March, April, May, and June, which causes much stress on human health and performance. The booming temperature requires more energy to be used in the act of cooling, a factor that puts pressure on the cities power supplies. Nevertheless, this season encourages or encourages specific agricultural practice or cultural practices within this region.

The core rainy or monsoon season begins in June and extends till September when the region receives steady rainfall due to monsoon showers. It is important for charging sources of water to replenish water sources and for supporting agriculture which is one of the major subsectors in the region's economy. But with monsoon also comes issues like flooding and waterlogging which the city cannot handle well now due to its outdated drainage system. These problems can affect mobility, trade and indeed every other activity within the society, which indicates why there is need for better city planning and infrastructure.

The cold season of winter is between the months of December and February, and although it is cold, the weather in Kolkata during this time of the year is perfect – ideal for improving the standard of living and for tourism. It is usually characterized by many festivals and out door activities that enhance the cultural aspect of the city.

However, over the past few years, socio-climactic change across the world has brought other uncertainties into the climatic mix of Kolkata. Climatic anomalies, the rising frequency of climatic erratic occurrences, minimal changes in climatic patterns, and gradual climatic shifts are considered conclusive evidence of climate change in the given area. These shifts have placed more pressure through the call for flexibility within the approaches to the planning of cities, the management of the environment and provision of policies.

It is past understanding, rather a necessity to decipher the multifaceted climatological circumstances of Kolkata. It must therefore be understood that preventive endeavours for instance enhancing infrastructure, encouraging sustainable development and involving the community are the initial steps to avoiding the negative impacts. These primary objectives have been grouped under two broad goals: Protecting our future – Promoting and encouraging the building of resilience and flexibility to ensure that the city can protect itself from the unknowns of a changing climate.

In conclusion, it could be acknowledged that Kolkata's climate is rather an active factor exerting a constant need for focused research and adjusting human activity. It means that with understanding the problems and utilizing the strengths of the weather, Kolkata can develop again in a sustainable manner. The triple helix constructive cooperation of governmental authorities, private companies, and people will play a crucial role in creating a city that sustains itself in extreme climate conditions prevailing there.

References

[1] Elin Lundstad, Yuri Brugnara, Duncan Pappert, Jérôme Kopp, Eric Samakinwa, André Hürzeler, Axel Andersson, Barbara Chimani, Richard Cornes, Gaston Demarée, Janusz Filipiak, Lydia Gates, Gemma L Ives, Julie M Jones, Sylvie Jourdain, Andrea Kiss, Sharon E Nicholson, Rajmund Przybylak, Philip Jones, Daniel Rousseau, Birger Tinz, Fernando S Rodrigo, Stefan Grab, Fernando Domínguez-Castro, Victoria Slonosky, Jason Cooper, Manola Brunet, and Stefan Brönnimann. The global historical climate database HCLIM. *Sci. Data*, 10(1):44, January 2023.

[2] Shanika Wickramasinghe. Leveraging Historical Weather Data for Climate Analysis — blog.weatherstack.com. `https://blog.weatherstack.com/blog/leveraging-historical-weather-data-for-climate-analysis/`, 2024.

[3] Arpan Manna. Weather_data_of_Kolkata — kaggle.com. `https://www.kaggle.com/datasets/kafkarps/five-years-weather-data-of-kolkata`, 2022.

[4] Autoregressive integrated moving average - Wikipedia — en.wikipedia.org. `https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average`.

[5] Subhro Paul. Complete Guide To SARIMAX in Python - GeeksforGeeks — geeksforgeeks.org. `https://www.geeksforgeeks.org/complete-guide-to-sarimax-in-python/`. [Accessed 11-11-2024].

[6] Liuyi Chen, Bocheng Han, Xuesong Wang, Jiazhen Zhao, Wenke Yang, and Zhengyi Yang. Machine learning methods in weather and climate applications: A survey. *Applied Sciences*, 13(21), 2023.

[7] Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Jason Hickey, Shreya Agrawal, and Nal Kalchbrenner. Metnet: A neural weather model for precipitation forecasting. *Submission to journal*, 2020.

[8] MySQL - Wikipedia — en.wikipedia.org. `https://en.wikipedia.org/wiki/MySQL`.