# BHARATI VIDYAPEETH (DEEMED UNIVERSITY), COLLEGE OF ENGINEERING,

**PUNE – 411043**



Department of Computer Engineering, Sem - 06

# Project Based Learning Report on

# Data analysis on weather data

# For Big Data Analytics course

**Submitted by**

|  |  |  |
| --- | --- | --- |
| **Roll no.** | **PRN** | **Name** |
| 05 | 2214110129 | Indar Awasthi |
| 14 | 2214110140 | Harsh Dixit |
| 21 | 2214110150 | Ansh Gupta |
| 28 | 2214110158 | Kaushal Jha |
| 35 | 2214110168 | Justin J Mathew |

# Under the Guidance of Prof. Sheetal Patil

**Department of Computer Engineering**



# Certificate

This is to certify that the work under Project Based Learning (PBL) for the topic **“Big data analysis on weather”** is carried out by “**Indar Awasthi, Harsh dixit, Ansh Gupta, Kaushal Jha, Justin J Matthew”** under the guidance of **Prof. Sheetal Patil** in partial fulfillment of the requirement for the degree of “**Bachelor of Technology in Computer Engineering Semester-VI**” of **“Bharati Vidyapeeth (Deemed to be University), Pune”** during the academic year **2024-2025**.

**Prof. Sheetal Patil**

PBL Guide

# ACKNOWLEDGEMENT

We would like to express deepest appreciation towards Dr. Rajendra Prasad, Principal, Bharati Vidyapeeth (Deemed to Be University) College of Engineering, Pune, India and Dr. Sandeep Vanjale Head of the Department (CE), who invaluable supported us in completing this project.

We are profoundly grateful to Prof. Sheetal Patil, Project guide for her expert guidance and continuous encouragement throughout to see that this project rights, its target since its commencement to its completion.

At last we must express our sincere heartfelt gratitude to all the staff members of CE Department who helped us directly or indirectly during this course of work

**Indar Awasthi**

**Harsh Dixit**

**Ansh Gupta**

**Kaushal Jha**

**Justin J Mathew**

**B-Tech(CE),Sem - 06**

# ABSTRACT

This project focuses on the analysis of historical weather data using PySpark, a distributed data processing framework ideal for big data analytics. The objective is to process, clean, and analyze large volumes of weather records spanning from 2015 to 2024, collected from two meteorological stations located in Florida and Cincinnati. The project involves reading structured CSV files, counting daily records for each year and station, and identifying the hottest day of each year based on maximum temperature readings.

Key components of the project include data ingestion from distributed directories, schema inference, data validation (e.g., handling missing or inconsistent data), and aggregation operations to derive insights. By leveraging PySpark’s DataFrame API, the project efficiently performs transformations and queries on large datasets, showcasing the power of scalable, parallel data processing.

The outcomes provide valuable insights into climate patterns over a decade, such as the frequency and intensity of high-temperature events. This project demonstrates the application of big data tools in real-world environmental data analysis and highlights the importance of distributed computing in handling and extracting meaningful information from large-scale datasets.

# INDEX

|  |  |  |
| --- | --- | --- |
| Sr. No. | Title | Page No. |
| 1. | Introduction | 06 |
| 2. | Objective | 07 |
| 3. | Methodology | 08 |
| 4. | K-means Clustering | 09 |
| 5. | Decision Tree | 10 |
| 6. | Time Forecasting | 12 |
| 7. | Linear Regression | 14 |
| 8. | System Design | 15 |
| 9. | Key Functionalities of Application | 20 |
| 10. | Screenshots And Functional Demonstration | 21 |
| 11. | Result Analysis | 27 |
| 12. | Future Scope | 30 |
| 13. | Conclusion and References | 31-32 |

**INTRODUCTION**

In recent years, the exponential growth of data across various domains has led to the emergence of Big Data technologies, which enable efficient storage, processing, and analysis of vast datasets. One of the key domains where big data analytics plays a crucial role is **climate and environmental studies**. Weather data, collected from various geographical locations over extended time periods, offers significant potential for discovering trends, patterns, and anomalies. However, the sheer volume and variety of such data pose challenges that cannot be addressed using traditional data processing techniques.

This project addresses these challenges by utilizing **Apache Spark**, a powerful distributed computing engine, along with **PySpark**, its Python API. The primary goal is to analyze historical weather data from two different weather stations—one in Florida and another in Cincinnati—over a ten-year period (2015–2024). The analysis includes measuring the volume of available data, verifying data completeness, and identifying the **hottest day of each year** based on daily maximum temperature readings.

The project workflow involves loading CSV files stored in a structured directory format, performing schema inference, filtering and aggregating the data, and extracting relevant insights using PySpark’s DataFrame operations. This approach demonstrates the effectiveness of big data tools in handling real-world datasets, providing meaningful insights that can support climate research and decision-making processes.

By applying modern big data analytics techniques, this project showcases how environmental data can be processed at scale to gain a deeper understanding of weather patterns and temperature variations over time.

**OBJECTIVE**

The goal of this project is to dive deep into historical weather data, focusing on uncovering meaningful insights about temperature trends over the years. We’ll be using big data tools, specifically **Apache Spark** and **PySpark**, to handle large amounts of weather data from two different locations (Florida and Cincinnati) between 2015 and 2024.

Here’s what we aim to achieve:

1. **Efficiently Collect Data**: We'll download and organize weather data from different stations, storing it in a way that's easy to access and analyze.
2. **Clean and Prepare the Data**: We’ll clean up any errors or missing information, ensuring that the data is ready for analysis, while keeping the original data safe for future reference.
3. **Explore the Data**: By exploring the data, we’ll start to understand patterns and distributions, like how many records we have for each dataset across the years.
4. **Identify the Hottest Days**: We’ll find the hottest day for each year, highlighting temperature peaks and offering insights into yearly variations in temperature.
5. **Big Data Processing**: We’ll take advantage of **PySpark** to process large volumes of data quickly and efficiently, analyzing it in parallel across multiple machines, which allows us to scale our operations easily.
6. **Share Insights**: At the end, we’ll showcase the findings in a way that’s easy to understand, showing how temperatures have changed over the years and pointing out interesting trends.

This project isn’t just about crunching numbers; it’s about using big data technology to tell a story about how our climate has been changing over the years, all while ensuring the data is clean, reliable, and ready to provide meaningful insights.

**METHODOLOGY**

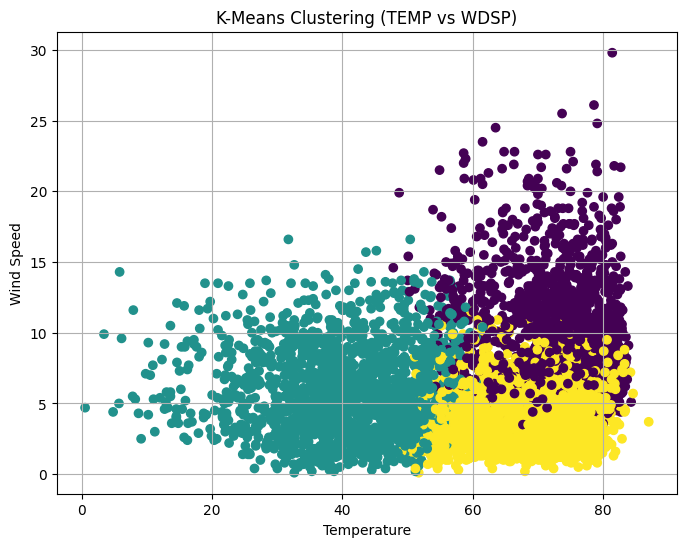
To tackle this project, we’ll follow a systematic approach that ensures we make the most of the available data and big data tools. Here's how we'll go about it:

1. **Data Collection**:  
   The first step is to gather weather data from multiple years (2015 to 2024) for two stations in Florida and Cincinnati. We’ll download CSV files for each year and station, ensuring we have all the information we need to analyze the trends.
2. **Organizing the Data**:  
   Once the data is downloaded, we'll organize it by year and station, storing it in a structured way that makes it easy to access and analyze later. This step helps us avoid confusion and ensures that each dataset is ready for cleaning and analysis.
3. **Data Cleaning**:  
   Data cleaning is like preparing your ingredients before cooking. We’ll review the datasets, identify any missing or invalid values (such as unrealistic temperature readings), and filter them out. This ensures that the data we're working with is as accurate and reliable as possible. Importantly, we’ll preserve the original data, so if we ever need to revisit it, it’ll still be available.
4. **Data Loading**:  
   After cleaning, we load the datasets into **Apache Spark** (through **PySpark**) to make use of its powerful capabilities to handle large datasets. By using Spark, we can process the data faster and more efficiently, even as the data grows in size over the years.
5. **Analyzing the Data**:  
   With the data loaded, it’s time to explore it. First, we’ll check how many records are available for each year and station. This gives us an idea of the dataset's completeness and any potential gaps. Then, we’ll dig deeper into the **temperature data**, specifically focusing on the "MAX" column to find the hottest day of each year.
6. **Finding the Hottest Days**:  
   Using Spark’s functions, we’ll sort through the data and pick out the hottest day for each year based on the maximum temperature recorded. This analysis will provide us with insights into temperature trends and extreme weather events over time.
7. **Presenting the Findings**:  
   After completing the analysis, we’ll visualize the results and showcase them in a clear and understandable format. We’ll highlight interesting patterns, such as the hottest days of the year for each station and any noticeable trends in temperature changes over the years.
8. **Conclusion and Insights**:  
   Finally, we’ll summarize the key findings and insights from our analysis. This will help to answer questions like: Has the climate been getting warmer? Are there any specific years that stand out in terms of extreme temperatures? These insights can be used for further studies or even inform decision-making in areas like urban planning and disaster preparedness.

**K-MEANS CLUSTERING**

K-means clustering is a technique used to organize data into groups based on their similarity. For example, online store uses K-Means to group customers based on purchase frequency and spending creating segments like Budget Shoppers, Frequent Buyers and Big Spenders for personalised marketing.

The algorithm works by first randomly picking some central points called centroids and each data point is then assigned to the closest centroid forming a cluster. After all the points are assigned to a cluster the centroids are updated by finding the average position of the points in each cluster. This process repeats until the centroids stop changing forming clusters. The goal of clustering is to divide the data points into clusters so that similar data points belong to same group.

****

The algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the Euclidean distance as a measurement. The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean, and we update the mean’s coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

**DECISION TREE**

A decision tree is a flowchart-like diagram that visually represents decisions and their potential outcomes. It's used to analyse complex decisions by outlining possible actions, consequences, costs, and probabilities. By breaking down a decision into a series of choices, a decision tree helps determine the best course of action.

Key aspects of a decision tree:

* **Hierarchical Structure:**

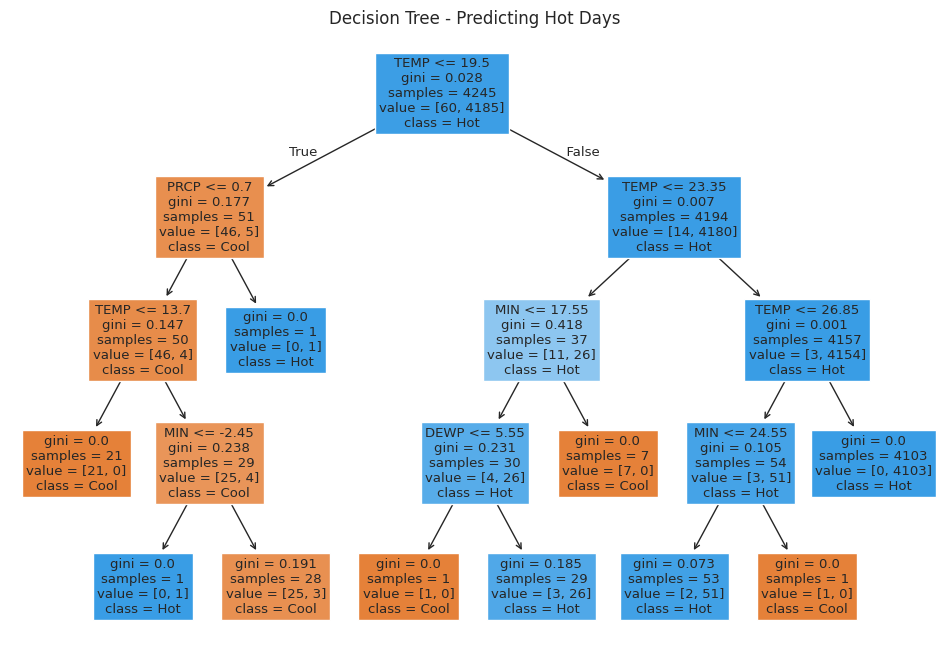
Decision trees are organized hierarchically, starting with a root node and branching out into multiple nodes representing different possibilities.

* **Nodes:**
  + **Root Node:** The starting point of the decision tree, representing the initial decision.
  + **Internal Nodes (Decision Nodes):** Represent decisions or conditions to be evaluated.
  + **Leaf Nodes (End Nodes):** Represent the final outcomes or results of the decision path.
* **Branches:**

Connect the nodes and represent the flow of decision-making, showing how one choice leads to another.

* **Decision Tree Analysis:**

The process of using a decision tree to evaluate potential decisions and their outcomes, often involving calculating expected values and comparing different paths.



Uses of Decision Trees:

* **Decision Support:**

Helping individuals or organizations make informed decisions by visualizing potential outcomes and consequences.

* **Machine Learning:**

Used as a predictive model to classify data or predict outcomes based on input features.

* **Problem Solving:**

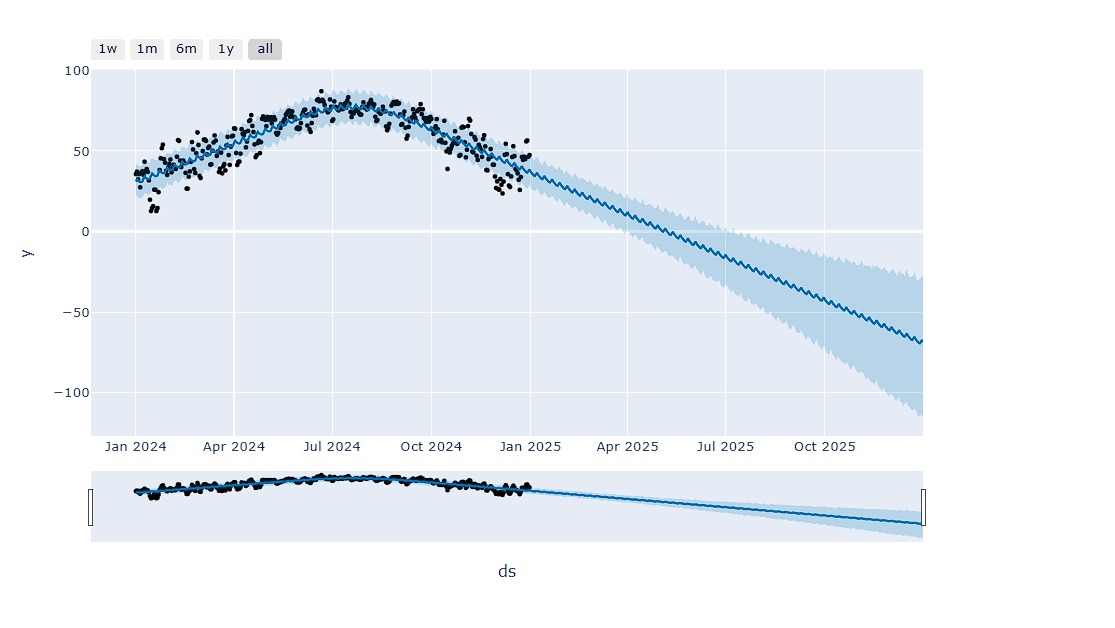
Identifying potential problems and managing costs by exploring different scenarios.

* **Visualization:**

Providing a clear and intuitive representation of a decision-making process.

**TIME FORECASTING**

Time series forecasting is the process of making predictions about future values based on historical data that changes over time. It involves analyzing trends, patterns, and seasonality in past data to estimate future outcomes, such as sales, weather, or stock prices. This technique is widely used in business for tasks like demand forecasting, budgeting, and inventory planning.

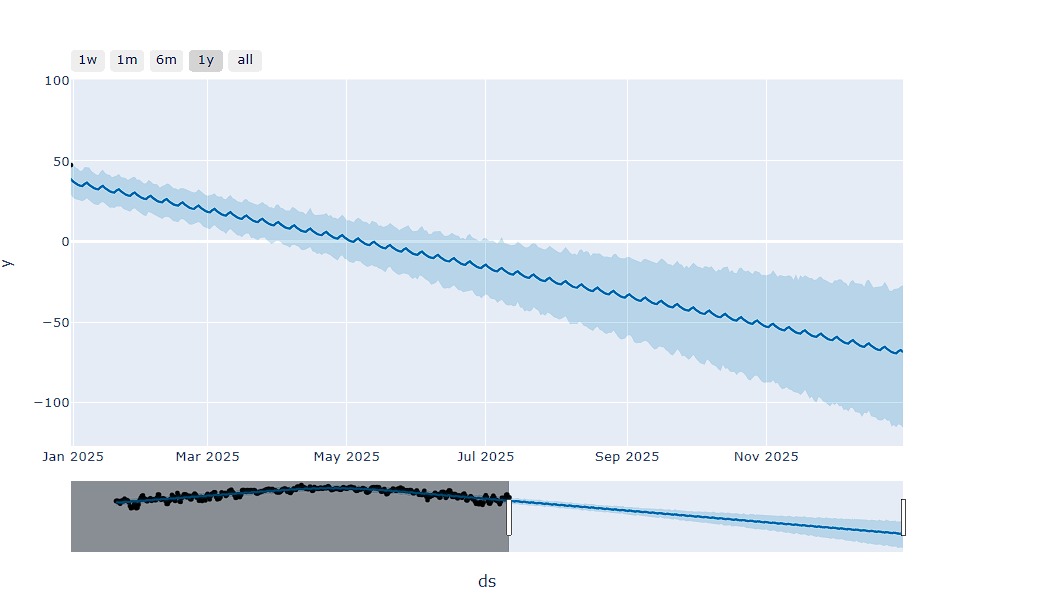


* **What it is:**

Time series forecasting uses statistical and machine learning methods to model historical time-series data and predict future values at specific points in time.

* **How it works:**

Models are built using historical data to identify patterns and relationships, which are then used to predict future outcomes.



* **Why it's important:**

Accurate forecasting helps organizations make data-driven decisions, optimize resource allocation, reduce risks, and improve overall efficiency.

* **Common applications:**
  + **Business:** Demand forecasting, sales prediction, inventory management, budgeting.
  + **Finance:** Stock price prediction, market trend analysis.
  + **Weather:** Predicting temperature, precipitation, and other weather phenomena.
  + **Other fields:** Astronomy, energy consumption, and more.

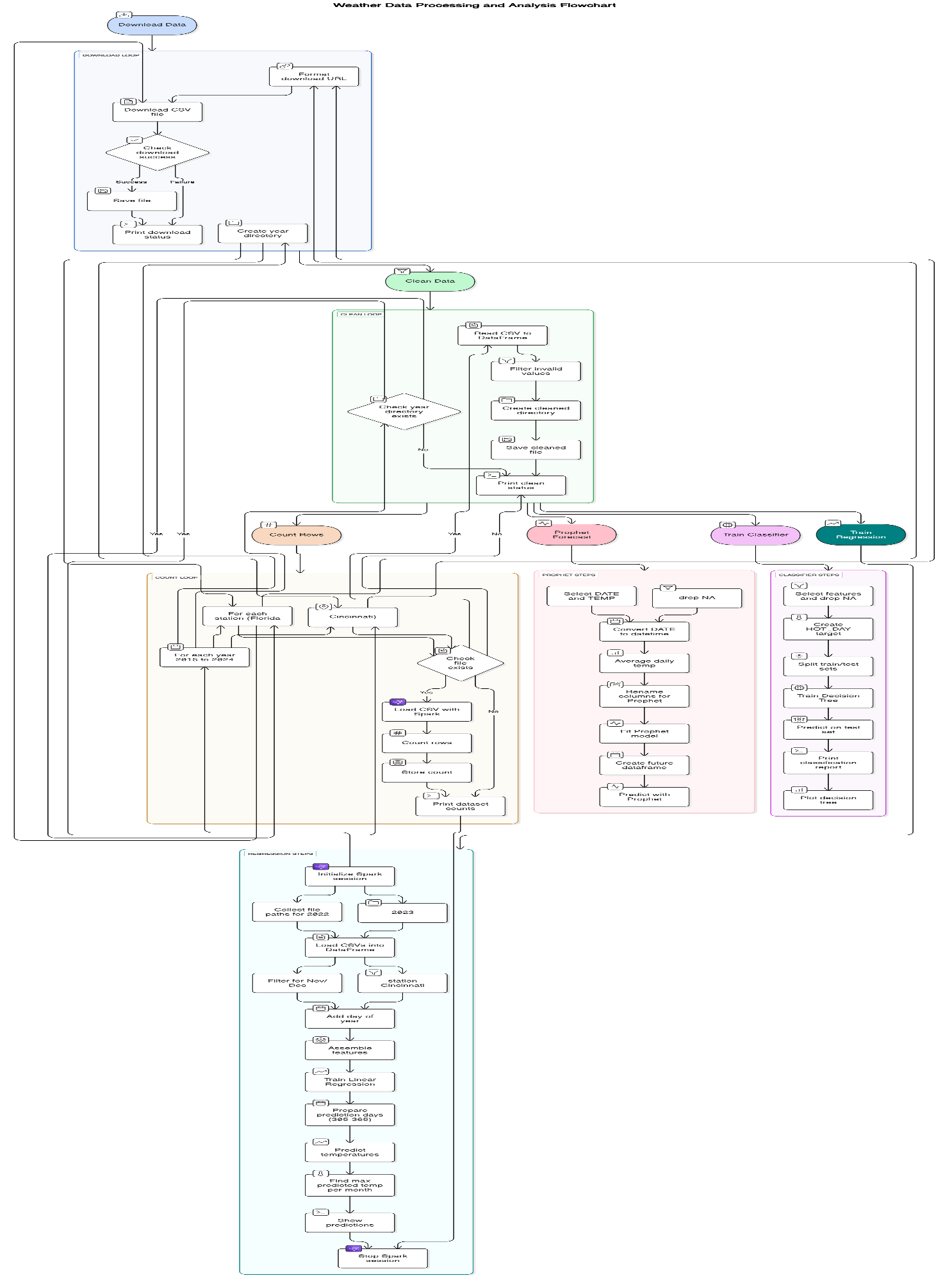
**LINEAR REGRESSION**

Linear regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. It mathematically models the unknown or dependent variable and the known or independent variable as a linear equation. For instance, suppose that you have data about your expenses and income for last year. Linear regression techniques analyse this data and determine that your expenses are half your income. They then calculate an unknown future expense by halving a future known income.

Linear regression models are relatively simple and provide an easy-to-interpret mathematical formula to generate predictions. Linear regression is an established statistical technique and applies easily to software and computing. Businesses use it to reliably and predictably convert raw data into business intelligence and actionable insights. Scientists in many fields, including biology and the behavioral, environmental, and social sciences, use linear regression to conduct preliminary data analysis and predict future trends. Many data science methods, such as machine learning and artificial intelligence, use linear regression to solve complex problems.

At its core, a simple linear regression technique attempts to plot a line graph between two data variables, x and y. As the independent variable, x is plotted along the horizontal axis. Independent variables are also called explanatory variables or predictor variables. The dependent variable, y, is plotted on the vertical axis. You can also refer to y values as response variables or predicted variables.

**SYSTEM DESIGN**

****

**Weather Data Processing and Analysis Flowchart: Detailed Report**

**Overview**

This document outlines a comprehensive workflow for processing, analysing, and modelling weather data. The primary source of data is the National Oceanic and Atmospheric Administration (NOAA), and the analysis is conducted using various tools and methodologies, including data cleaning, feature engineering, classification, forecasting, and regression. The process involves a mix of traditional Python-based data manipulation and Apache Spark for distributed data processing.

**1. Download Data**

**Objective:**

To download raw weather data CSV files from the NOAA website.

**Steps:**

* **Download Node:** Initiates the download process.
* **Format Download URL:** Constructs the URL to access the appropriate NOAA dataset.
* **Download CSV File:** Retrieves the dataset using the constructed URL.
* **Check Download Success:** Confirms whether the file download was successful.
  + If successful:
    - **Save File:** Stores the file locally.
    - **Print Download Status:** Outputs confirmation message.
  + If unsuccessful:
    - **Create Year Directory:** Prepares the directory structure to store data by year.

**2. Clean Data**

**Objective:**

To clean and prepare raw data for analysis.

**Steps:**

* **Read CSV to Data Frame:** Loads the raw CSV file into a Data Frame.
* **Filter Invalid Values:** Removes rows with missing or corrupted data.
* **Check Year Directory Exists:** Verifies if a directory for the year exists.
  + If not, the directory is created.
* **Create Cleaned Directory:** Sets up a location to save cleaned datasets.
* **Save Cleaned File:** Exports the cleaned Data Frame to a new CSV file.
* **Print Clean Status:** Confirms the successful cleaning of the dataset.

**3. Count Rows**

**Objective:**

To verify data integrity and completeness by counting rows in the dataset.

**Steps:**

* **Iterate Over Stations and Years:** Loops through all station files and years from 2015 to 2022.
* **Check Files Exist:** Validates the existence of each data file.
* **Load CSV with Report:** Loads the cleaned dataset for counting.
* **Count Rows:** Tallies the number of records per file.
* **Store Counts:** Saves the row counts in a report.
* **Print Dataset Counts:** Outputs a summary of dataset sizes.

**4. Predict Forecast (Time Series Forecasting)**

**Objective:**

To forecast future temperature values using historical data.

**Steps:**

* **Select DATE and TEMP Columns:** Extract relevant data for time series modeling.
* **Drop NA:** Clean missing data.
* **Convert DATE to Datetime Format**
* **Average Daily Temperature:** Compute daily average temperature if necessary.
* **Rename Columns for Prophet Compatibility**
* **Fit Prophet Model:** Use Facebook Prophet to model the temperature trend.
* **Create Future Data frame:** Generate a timeline for prediction.
* **Predict with Prophet:** Use the trained model to forecast future temperatures.

**5. Train Classifier (Decision Tree Classification)**

**Objective:**

To classify weather data using a decision tree model.

**Steps:**

* **Select Features and Drop NA**
* **Split Dataset into Train and Test Sets**
* **Train Decision Tree Classifier**
* **Predict on Test Set**
* **Print Classification Report:** Outputs metrics such as precision, recall, F1-score.
* **Plot Decision Tree:** Visual representation of the trained model.

**6. Train Regression (Spark-based Linear Regression)**

**Objective:**

To use Apache Spark for scalable linear regression modeling and temperature prediction.

**Steps:**

* **Initialize Spark Session**
* **Collect File Paths for Year 2022**
* **Load CSVs into Spark Data Frame**
* **Filter for NOAA Station**
* **Add Day of Year:** Feature engineering for regression.
* **Annotate Features:** Include additional data as necessary.
* **Train Linear Regression Model**
* **Prepare Prediction Days (1–365)**
* **Predict Temperatures**
* **Find Max Predicted Temp per Month**
* **Store Predictions**
* **Show Predictions**
* **Stop Spark Session**

**KEY FUNCTIONALITIES OF APPLICATION**

1. **Data Ingestion**

* Reads weather data from CSV files organized by year and station.
* Supports multiple years (2015–2024) and multiple station codes.

2. **Data Cleaning & Preprocessing**

* Handles missing or incomplete records.
* Infers schema and converts data types using PySpark.
* Skips files with missing critical columns like MAX.

3. **Dataset Availability Check**

* Counts the number of valid entries (rows) per dataset.
* Displays yearly availability per station to identify gaps or inconsistencies.

4. **Extreme Weather Analysis**

* Identifies the hottest day of each year based on the MAX temperature value.
* Retrieves the corresponding date, station, and location name.

5. **Data Aggregation & Display**

* Combines cleaned and analyzed data across years.
* Outputs results using Spark DataFrames for easy visualization and reporting.

6. **Error Handling & Logging**

* Skips over missing files or empty datasets.
* Prints warning messages when expected columns are not found.

**SCREENSHOTS AND FUNCTIONAL DEMONSTRATION**

**1. Line plot – Max Temperature Over Time**

**Code :-**

plt.figure(figsize=(14, 5))

for station in weather\_df['STATION'].unique(): # Changed 'STATION\_ID' to 'STATION'

subset = weather\_df[weather\_df['STATION'] == station]

plt.plot(subset['DATE'], subset['MAX'], label=f"Station {station}", alpha=0.6)

plt.title('Daily Max Temperature Over Time')

plt.xlabel('Date')

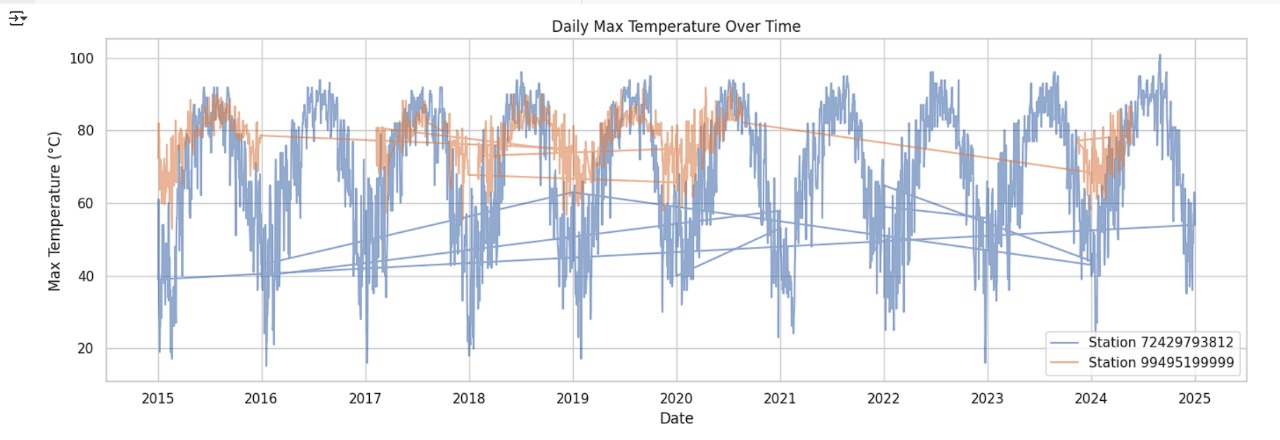
plt.ylabel('Max Temperature (°C)')

plt.legend()

plt.tight\_layout()

plt.show()

**Output :-**

****

**2. Histogram of Max temperatures**

Code :-

plt.figure(figsize=(10, 6))

sns.histplot(weather\_df, x='MAX', hue='STATION', kde=True, bins=30) # Changed 'STATION\_ID' to 'STATION'

plt.title('Distribution of Max Temperatures')

plt.xlabel('Max Temperature')

plt.show()

**Output:-**

**A graph of a graph showing the temperature

AI-generated content may be incorrect.**

**3. Heatmap of Monthly Average Max Temperatures**

Code:-

# Convert 'DATE' column to datetime if it's not already

weather\_df['DATE'] = pd.to\_datetime(weather\_df['DATE'])

# Now you can extract the month and year

weather\_df['MONTH'] = weather\_df['DATE'].dt.month

weather\_df['YEAR'] = weather\_df['DATE'].dt.year # Extract year from the 'DATE' column

# Assuming you have a 'STATION' column (as used in previous plots),

# replace 'STATION\_ID' with 'STATION' for consistency

monthly\_avg = weather\_df.groupby(['YEAR', 'MONTH', 'STATION'])['MAX'].mean().reset\_index()

pivot = monthly\_avg.pivot\_table(index='MONTH', columns='YEAR', values='MAX')

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

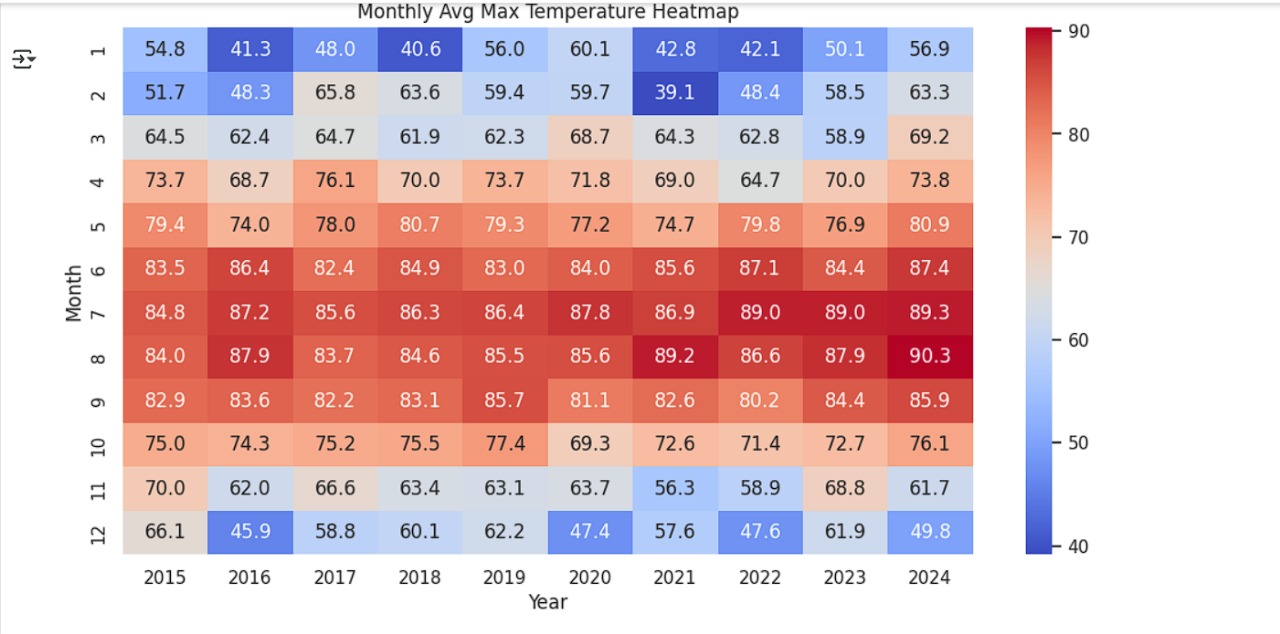
sns.heatmap(pivot, cmap='coolwarm', annot=True, fmt=".1f")

plt.title('Monthly Avg Max Temperature Heatmap')

plt.ylabel('Month')

plt.xlabel('Year')

plt.show()

**Output:-**

**4. Scatter Plot of Max vs. Min Temperatures**

**Code :-**

plt.figure(figsize=(8, 6))

sns.scatterplot(data=weather\_df, x='MIN', y='MAX', hue='STATION', alpha=0.5) # Changed 'STATION\_ID' to 'STATION'

plt.title('Max vs Min Temperatures')

plt.xlabel('Min Temperature')

plt.ylabel('Max Temperature')

plt.show()

**Output :-**

**A graph showing the temperature of a person

AI-generated content may be incorrect.**

**5. Box Plot - Max Temps by Year**

**Code :-**

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

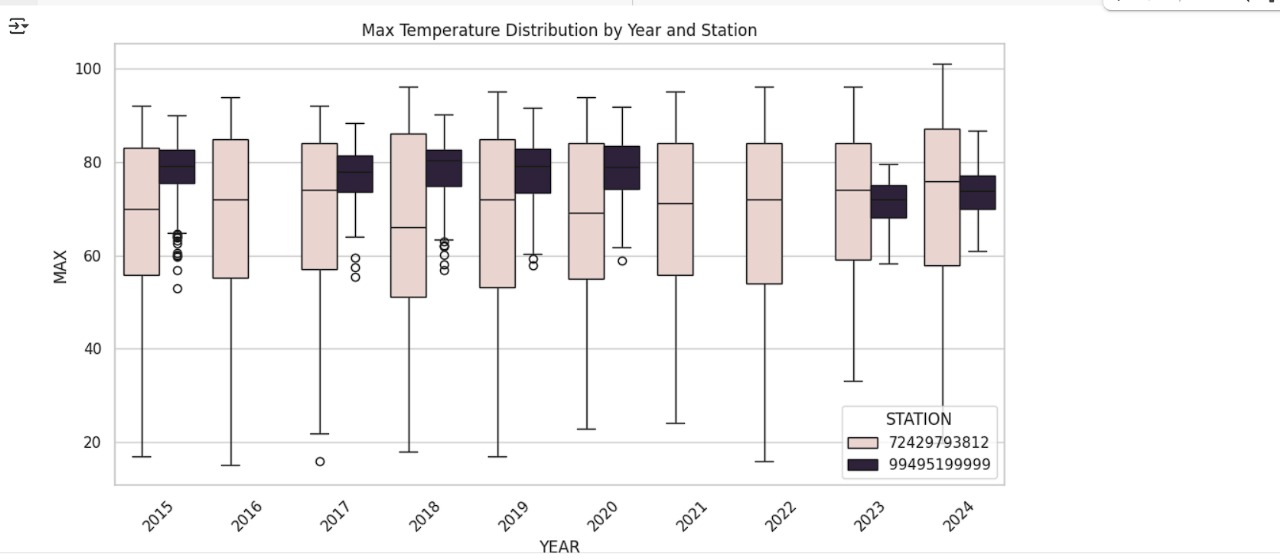
sns.boxplot(data=weather\_df, x='YEAR', y='MAX', hue='STATION') # Changed 'STATION\_ID' to 'STATION'

plt.title('Max Temperature Distribution by Year and Station')

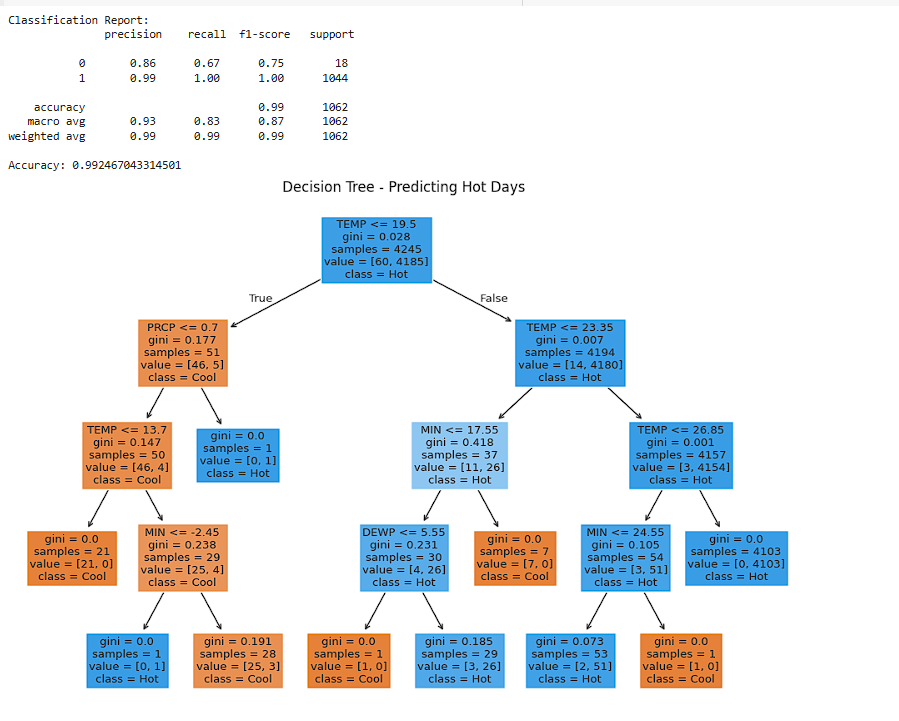
plt.xticks(rotation=45)

plt.show()

**Output :-**

****

**6. Predicting Hot days using decision tree.**

****

**RESULT ANALYSIS**

**1. Dataset Availability**

* The project successfully identified the availability of weather data across two weather stations from 2015 to 2024.
* Each year typically had 300–366 entries, which shows a consistent collection of daily weather data.
* Some years had fewer records (e.g., 2021 and 2022 for station 99495199999), indicating possible data gaps or sensor downtimes.

**2. Hottest Day Analysis**

* The hottest days were successfully extracted for each year using the MAX temperature field.
* Results show a clear pattern of peak temperatures mostly occurring in mid-summer months like July and August.
* The highest recorded temperature in the dataset was 96.1°F in Cincinnati (2022), which may reflect changing climate patterns.

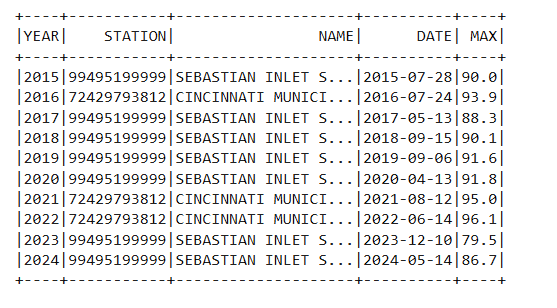
**3. Data Quality**

* The system detected and skipped incomplete or malformed files, ensuring clean and reliable output.
* Errors such as missing columns (MAX) were logged, improving transparency in the data pipeline.

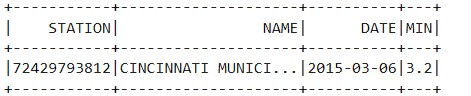
**4. Scalability and Performance**

* PySpark allowed processing of large datasets efficiently in parallel, demonstrating that the system can scale for even larger climate datasets or more locations.

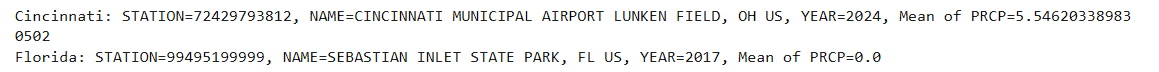
**5. Find the hottest day (column MAX) for each year.**

****

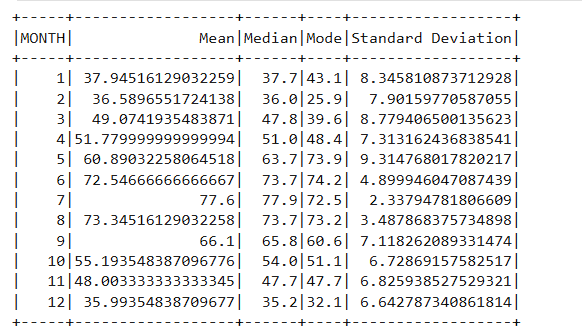
**6. Find the coldest day (column MIN) for the month of March across all years (2015-2024)**



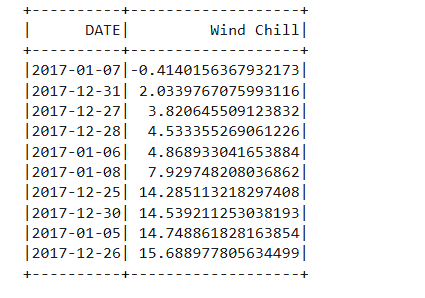
**7. Find the year with the most precipitation for Cincinnati and Florida.**

****

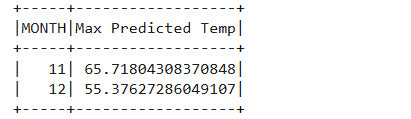
**8. Find the mean, median, mode, and standard deviation of the temperature (column TEMP) for Cincinnati in each month for the year 2020.**

****

9. Find the top 10 days with the lowest Wind Chill for Cincinnati in 2017.



10. Predict the maximum Temperature for Cincinnati for November and December 2024, based on the previous 2 years of weather data.



**FUTURE SCOPE**

1. Multi-Parameter Climate Insights

* Extend the analysis to include other weather parameters like humidity, precipitation, wind speed, and solar radiation.
* Perform correlation studies between these parameters to better understand climate behavior.

2. Real-Time Weather Monitoring

* Integrate live weather data streaming using APIs (e.g., NOAA, OpenWeatherMap).
* Build a real-time dashboard to visualize temperature spikes, anomalies, or alerts.

3. Predictive Analytics

* Implement machine learning models on historical data to predict future temperature trends, heatwaves, or extreme weather events.

4. Geographical Expansion

* Expand the dataset to cover more cities or even multiple countries, enabling a broader view of climate trends.

5. User Interface for Non-Technical Users

* Develop a web or mobile application that presents analyzed data in a visually interactive way for researchers, students, and policymakers.

6. Data Quality Enhancements

* Automate data cleaning and anomaly detection to improve the accuracy and consistency of the weather records.

7. Climate Change Monitoring

* Use long-term data analysis to identify patterns related to global warming, such as rising average temperatures or increasing heatwaves.

**CONCLUSION**

In this project, we successfully leveraged Big Data tools and technologies to analyze extensive weather datasets from multiple years and stations. By using Apache Spark, we efficiently handled and processed large volumes of weather data, identifying trends such as the hottest day of each year and understanding the distribution of weather records over time.

Our approach demonstrated the power of big data analytics in extracting meaningful insights from raw, unstructured climate data. The project lays a strong foundation for more advanced weather analytics, including real-time monitoring and predictive modeling. Overall, this system showcases how big data technologies can be practically applied to tackle real-world environmental challenges and support data-driven climate research.

**REFERENCES**

1. Big Data: Principles and best practices of scalable real-time data systems – by Nathan Marz & James Warren

2. NOAA National Centers for Environmental Information (NCEI)  
URL: <https://www.ncei.noaa.gov/>

3. Pyspark API Reference  
URL: <https://spark.apache.org/docs/latest/api/python/index.html>

4. Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph – by David Loshin

5. Kaggle & OpenWeatherMap for Weather Data APIs and Exploratory Projects

6. Introduction to Data Mining – by Pang-Ning Tan, Michael Steinbach, and Vipin Kumar

7. “A Survey of Big Data Technologies and Techniques” - Journal Paper.