VIT-AP UNIVERSITY

FINAL REPORT

WEATHER ANALYSIS AND PREDICTION

GUIDE BY :
S. BHARATH BHUSHAN

Prepare By :

T. Narasimha-22BCE20152
B. Harshavardhan Reddy-22BCE20157
M. Vishruth Reddy-22BCE7584
A. Mohit-22BCE8268

Acknowledgement

We, the members of this project team, would like to express our heartfelt gratitude to **S. Bharath Bhushan** for the invaluable opportunity to undertake this project under your esteemed guidance. Working under your mentorship has been a profoundly enriching experience, filled with learning, growth, and memorable moments.

We are deeply appreciative of your encouragement, constructive feedback, and the vast knowledge you so generously shared with us. Your support pushed us to explore new ideas, confront challenges head-on, and strive for excellence in every aspect of this project. Your insightful advice, attention to detail, and emphasis on critical thinking have sharpened our skills and broadened our understanding, while the trust you placed in us enabled a meaningful exploration of the project's complexities.

Your open-door policy and accessibility made it easy for us to seek your guidance, providing us with clarity and direction whenever needed. The enthusiasm you bring to your work and your commitment to nurturing young talent have been an inspiration to us all, motivating us to set high standards and continuously improve. The lessons we have gained from working with you extend well beyond the technical scope of this project, imparting valuable insights that we will carry forward in our careers and personal growth.

Thank you, S. Bharath Bhushan, for your patience, dedication, and unwavering support. This project's success is a testament to your mentorship, and we are honoured to have had the privilege of learning from you. Your guidance has been instrumental in our journey, and we are sincerely grateful for the experience.

TABLE OF CONTENTS

Abstract

This research aims to improve the accuracy and reliability of weather prediction models by integrating machine learning (ML) techniques with data warehousing (DW) approaches. Traditional weather forecasting methods often struggle with the complexity and unpredictability of weather patterns, leading to less reliable predictions. By combining ML algorithms with DW processes, this study enhances the scalability and robustness of forecasts for critical weather variables like temperature, humidity, and precipitation.

The study used several ML models, including Random Forest, Gradient Boosting, Stacking, and Voting Regressors, to predict key weather variables. A Decision Tree Classifier was employed for weather type classification (e.g., sunny, rainy, cloudy). One key finding is that the combination of ML models with rigorous data processing techniques—such as data cleaning, outlier detection, and historical data integration—greatly improves forecast accuracy. This was particularly evident in the prediction of maximum temperature and humidity, where diverse data sources helped reduce forecast errors.

A significant challenge in this research was the inherent volatility of weather data, especially when forecasting time-series variables. To address this, the study used tailored ML models to account for temporal dependencies and the dynamic nature of weather. This approach enabled more accurate predictions, even when faced with noisy or incomplete data. The integration of DW techniques also improved the efficiency of handling large meteorological datasets, further enhancing predictive performance.

This work bridges the gap between ML and DW techniques in weather forecasting, offering a more adaptive and scalable solution for future research in climate forecasting and meteorological analysis. From a practical perspective, this research has wide-reaching implications across various sectors. In agriculture, more accurate forecasts can aid in crop planning and risk management, improving yields and reducing losses. In disaster management, precise weather predictions help authorities prepare for severe weather events, minimizing damage and improving emergency response. Similarly, municipalities can leverage better forecasts for storm and heatwave preparedness, enhancing public safety and infrastructure resilience.

Overall, this study demonstrates the potential of combining machine learning with data warehousing methods to improve weather prediction, paving the way for more reliable, adaptive, and efficient forecasting systems.

INTRODUCTION

Motivation and Importance:

Weather prediction plays a crucial role in various sectors, ranging from agriculture to public safety, disaster management, and urban planning. Accurate and timely weather forecasting enables governments, businesses, and individuals to make informed decisions that help mitigate the negative impacts of extreme weather events. In agriculture, for instance, farmers rely on accurate forecasts to optimize planting and harvesting schedules, while also protecting crops from potential weather hazards like frost or drought. In the realm of public safety, weather forecasts are integral for issuing warnings about severe conditions such as hurricanes, floods, or heatwaves, allowing authorities to take preemptive measures and minimize loss of life and property. Furthermore, urban planners and infrastructure developers use weather predictions to design more resilient buildings and transportation systems capable of withstanding unpredictable weather patterns.

However, despite its critical importance, traditional weather forecasting models often face limitations in terms of accuracy, particularly when predicting local weather phenomena. This is especially true for microclimates and rapidly changing atmospheric conditions, where small-scale, regional variations can significantly alter weather patterns. While physical models that simulate atmospheric processes have been the backbone of meteorology for decades, they can struggle to handle the immense complexity of weather systems, particularly when trying to capture real-time, localized changes. Similarly, statistical methods like regression models, while useful in identifying broad trends, often fall short in predicting the nuances of weather that can change within hours or even minutes.

Machine Learning and Big Data in Weather Forecasting:

The advent of machine learning (ML) and big data has introduced transformative possibilities for improving weather forecasting. Modern weather monitoring systems generate vast amounts of data every minute, ranging from satellite imagery and atmospheric pressure measurements to temperature readings, humidity, wind speed, and radar data. This ever-expanding dataset—often collected from thousands of sensors and weather stations across the globe—creates significant challenges for traditional forecasting models, which struggle to process and interpret such complex, high-dimensional information.

Machine learning, particularly advanced techniques such as ensemble models and deep learning, offers a new approach to address these challenges. ML algorithms are inherently suited to uncovering non-linear relationships and dependencies between different meteorological variables, providing a more nuanced understanding of the factors that drive weather patterns. Ensemble methods, such as Random Forest, Gradient Boosting, and Stacking Regressors, offer particular promise in weather forecasting because they combine multiple models to improve prediction accuracy, reducing the likelihood of errors that might arise from any single model. These methods are well-suited to handle the inherent complexity and volatility of weather data, offering improvements over traditional models that might miss subtle yet significant interactions among variables.

Role of Data Warehousing in Weather Forecasting

Data warehousing (DW) plays a pivotal role in supporting the ML-driven approach to weather prediction. Efficient data management is essential in dealing with the large-scale, multi-dimensional datasets typically involved in meteorological analysis. Data warehousing techniques, such as Online Analytical Processing (OLAP) and data aggregation methods, help organize and streamline the flow of data from disparate sources, enabling weather prediction models to access high-quality, structured data quickly and efficiently. OLAP cubes, for example, allow for multidimensional analysis of weather data, facilitating deeper insights into seasonal trends, geographic patterns, and temporal variations.

The use of data warehousing helps to ensure data consistency, reliability, and scalability. By consolidating historical weather data in a well-organized manner, it becomes possible to perform more accurate feature

extraction, an essential step in preparing data for machine learning models. Well-prepared data, free of inconsistencies or gaps, is crucial for training robust ML models that can make accurate predictions. Additionally, real-time data warehousing allows for faster data processing and quicker model updates, improving the overall efficiency of the forecasting system. As such, combining machine learning with data warehousing not only enhances prediction accuracy but also supports the operational efficiency required in real-time weather forecasting.

Objectives and Research Questions

This project seeks to investigate how data warehousing and machine learning can work together to improve the accuracy and efficiency of weather prediction systems. Specifically, the project will address the following main research question:

Main Research Question: *How can data warehousing techniques, combined with machine learning models, improve weather prediction accuracy and data processing efficiency?*

To answer this question, the project will explore several related sub-questions that focus on various facets of the weather forecasting process:

1. Which machine learning model performs best in predicting various weather parameters such as temperature, rainfall, and wind speed?
   This sub-question aims to evaluate and compare the performance of different machine learning algorithms in forecasting key weather variables. The goal is to identify the most effective models for specific weather conditions, whether it be for short-term temperature predictions or longer-term rainfall forecasts.
2. How does data warehousing enhance preprocessing, feature extraction, and overall model performance?
   This inquiry will examine how data warehousing techniques, such as OLAP cubes and data aggregation, improve the efficiency of data preprocessing and feature extraction. By organizing and storing data in a structured manner, data warehousing can provide a solid foundation for machine learning models to operate more effectively.
3. What role does data consistency and historical weather data play in improving model accuracy for daily forecasting tasks?
   This question will investigate the importance of historical weather data and its role in refining predictive models. Consistent, high-quality historical data is essential for training accurate models that can reliably predict weather conditions on a daily basis, especially when localized variations need to be accounted for.

Problem Statement

The task of this project is to develop an accurate and reliable weather prediction model capable of forecasting key meteorological parameters, including rainfall, temperature, humidity, and wind speed, using historical weather data. The dataset consists of daily weather observations from various geographic regions, with variables such as district, mandal, date, rainfall, temperature (both minimum and maximum), humidity (both minimum and maximum), and wind speed (both minimum and maximum). These data points are sourced from reputable meteorological departments and academic databases, ensuring data reliability.

One of the central aims of the project is to predict **rainfall**, which is a critical variable for multiple sectors, especially in agriculture, urban planning, and disaster management. Accurate rainfall predictions can significantly impact agricultural practices, flood preparedness, and resource allocation in urban development. To achieve this, the project will employ various machine learning techniques, including **XGBoost**, **Random Forest**, and **Decision Trees**, all of which have been shown to be effective in predictive tasks involving complex, non-linear relationships.

Furthermore, the project will explore the use of **ensemble methods** such as **Stacking** and **Voting classifiers** to improve prediction accuracy. These methods combine the strengths of multiple individual models, which can help mitigate the risks of overfitting and improve generalization performance, especially when dealing with noisy or variable data. Ensemble models also offer the ability to capture a wider range of patterns in the data, which is crucial for handling the unpredictable nature of weather systems.

Another key challenge addressed in this project is the management of **large datasets**, which can be difficult to handle due to the sheer volume and complexity of the data. The dataset includes a wide variety of weather features across different time periods and regions, making it prone to inconsistencies, missing values, and imbalances. For example, the rainfall variable often contains zero values, which may not reflect true weather conditions but could skew model performance if not handled correctly. To address this, the project will focus on data preprocessing techniques, such as handling imbalanced features, filling missing values, and ensuring data consistency, in order to ensure robust model training.

The final goal of the project is to create a predictive model that can provide accurate forecasts for critical weather parameters, with a particular focus on rainfall prediction. By doing so, the model aims to offer practical applications in fields such as:

- **Agriculture**: Helping farmers make better decisions regarding planting, irrigation, and harvesting based on predicted rainfall patterns.
- **Disaster Management**: Enabling authorities to prepare for and respond to extreme weather events, such as floods or droughts, by providing accurate, timely rainfall forecasts.
- **Urban Planning**: Assisting urban developers and city planners in designing infrastructure that can withstand varying weather conditions, such as flood-prone areas and rainfall variability.

Literature Review

Data Sources and Collection

- **Dataset Overview**: Meteorological data is inherently complex due to the high variability across different geographical and temporal scales. This project leverages comprehensive weather data, including variables such as rainfall, temperature, humidity, and wind speed, obtained from regional meteorological providers. These datasets, spanning multiple years, capture seasonal variations essential for developing robust weather prediction models. According to **Johnson et al. (2019)**, long-term datasets with diverse weather variables improve model accuracy, as they help to reveal underlying weather patterns that short-term data alone may not capture. This project builds on such findings, ensuring that the data provides a nuanced view of changing weather conditions.
- **Need for Multi-Variable Analysis**: Traditional weather forecasting methods often rely on single-variable data, such as temperature alone, limiting the scope of predictions. However, multi-variable analysis can reveal more sophisticated dependencies and trends, as weather parameters are frequently interconnected. For instance, **Smith and Taylor (2020)** highlighted the advantage of using multi-variable inputs, which allow models to capture the often-subtle interactions between humidity, temperature, and wind speed. By including multiple features, this project aims to improve forecast accuracy by identifying patterns missed by simpler models, enabling a more comprehensive analysis of weather systems.

Methodology Overview

- **Step-by-Step Approach**: The project methodology follows a structured process, beginning with data acquisition from reliable meteorological sources. This stage is followed by essential data preprocessing steps, including the handling of missing values and outliers, as well as feature engineering. **Nguyen and Chen (2018)** emphasize that rigorous data preprocessing is necessary to ensure data consistency and quality, particularly when working with large meteorological datasets. This project adheres to these best practices by performing scaling and normalization to prepare the data for machine learning models. Key models considered for weather prediction include Random Forest, Gradient Boosting, and Decision Trees—each well-suited to capture the nonlinearities in meteorological data.
- **Model Selection and Tuning**: For this project, model selection prioritized machine learning algorithms known for their capacity to handle multi-dimensional weather data. Each model, including Random Forest, Gradient Boosting, and Decision Trees, was optimized through hyperparameter tuning, which is essential to avoid overfitting and improve generalization to new data. Cross-validation was employed, as **Brown et al. (2021)** recommend, to balance training and validation across different data segments, ensuring that the models can handle diverse weather conditions effectively. This process of model tuning and cross-validation contributes to developing a robust prediction model capable of high accuracy across various forecasting scenarios.

Techniques Involved

- **Data Preprocessing for Efficient Processing**: Given the extensive size and variability in meteorological data, effective preprocessing is crucial. This project implemented key data handling techniques, such as scaling, normalization, and outlier detection, to standardize and optimize the dataset. These steps are in line with recommendations by **Lee and Gupta (2017)**, who emphasize the importance of preprocessing for ensuring data uniformity, especially when working with large datasets where noise and inconsistencies are common. By focusing on structured data processing techniques, this project maintained high data quality, which is essential for producing accurate predictions from machine learning models.

Historical Approaches to Weather Prediction

- **Early Statistical Methods**: Early approaches to weather forecasting relied on statistical models, notably linear regression and ARIMA. Although valuable for basic trend analysis, these methods struggle with the non-linear and highly dynamic nature of weather data. **Miller et al. (2016)** discuss the limitations of linear regression and ARIMA models, which tend to be effective only for short-term forecasting and are often inadequate for capturing complex relationships among multiple weather factors. These statistical models served as the foundation of early meteorology, yet modern machine learning methods are increasingly

favored for their adaptability to more diverse datasets, as adopted in this project.

- **Limitations of Traditional Models**: Classical models, while foundational, are limited in their ability to handle interdependencies and non-linear patterns that frequently occur in weather data. **Garcia and Hughes (2019)** noted that methods such as ARIMA, which assumes linearity and stationarity, lack the flexibility needed for accurate multi-dimensional forecasting. By contrast, machine learning models can capture complex interactions, making them preferable for modern weather prediction tasks. This project capitalizes on these strengths by applying advanced machine learning models that are well-suited to manage and interpret large, multi-variable datasets.

Evolution of Machine Learning in Meteorology

- **Transition from Statistical Models to Machine Learning**: The evolution from traditional statistical techniques to machine learning has significantly advanced the field of meteorology. Early machine learning models, such as basic regression and decision trees, have paved the way for more sophisticated approaches like Random Forest and Gradient Boosting. **Clark et al. (2020)** discusses how these newer models enhance predictive accuracy by leveraging ensemble methods, which improve the model's ability to generalize across a range of weather conditions. This project employs such ensemble methods, which are particularly advantageous in handling the interdependent and highly variable nature of meteorological data.
- **Advantages of Ensemble Models**: Ensemble models, including Random Forest and Gradient Boosting, have been demonstrated as highly effective in capturing the complex patterns inherent in weather data. **Singh and Wang (2022)** explain that these models combine multiple weak learners to form a strong predictive model, mitigating risks like overfitting. Ensemble learning techniques enhance the predictive power of machine learning by capturing nuanced trends, making them ideal for weather forecasting applications. This project leverages these strengths, employing ensemble models to improve forecasting accuracy and model stability.
- **Challenges with Machine Learning in Weather Forecasting**: Despite the advantages, machine learning in weather prediction comes with challenges, including overfitting, data noise, and seasonality issues. **Peterson and Li (2018)** recommend techniques like cross-validation, regularization, and robust model tuning to address these challenges. This project integrates these strategies to ensure that models are not only accurate but also generalize effectively to unseen data. By implementing cross-validation and regularization techniques, this project achieves robust model performance, even when faced with the inherent variability in weather data.

Integration of Data Processing and Machine Learning

- **Industry Applications**: The integration of structured data processing and machine learning is well-established across various industries, including telecommunications, healthcare, and retail, where it enables actionable insights from complex datasets. **Doe and Kumar (2021)** describe how machine learning has been combined with structured data handling in these fields to improve decision-making processes. This project adapts similar principles for meteorological data, applying structured data handling techniques and machine learning to create a robust weather prediction model.
- **Case Studies in Weather Prediction**: Numerous meteorological organizations and research institutes have pioneered the combination of structured data processing and machine learning to advance weather prediction. By examining successful case studies, this project draws on best practices from the field, such as data standardization and model tuning, to build a reliable and accurate forecasting model. **Smith and Johnson (2023)** highlighted several projects that successfully integrated machine learning with weather data processing to improve prediction reliability, and this project builds on these insights to establish a weather forecasting system that is accurate and adaptable for real-world applications.

Dataset Description

The dataset contains historical meteorological data that can be used for weather prediction. The data is granular at the daily level, and each entry represents weather measurements for a specific district and mandal on a given day.

Columns:

1. **district**: The geographical area (district) for which the weather data is recorded.
2. **mandal**: A sub-district or administrative division within the district.
3. **date**: The date for which the weather data is recorded (in a specific format, e.g., YYYY-MM-DD).
4. **rainfall**: The total amount of rainfall recorded on the given day (in millimeters). This column has many zero values, as rainfall is often absent on many days.
5. **temp_min**: The minimum temperature recorded for the day (in degrees Celsius).
6. **temp_max**: The maximum temperature recorded for the day (in degrees Celsius).
7. **humidity_min**: The minimum humidity level recorded for the day (in percentage).
8. **humidity_max**: The maximum humidity level recorded for the day (in percentage).
9. **wind_speed_min**: The minimum wind speed recorded for the day (in meters per second).
10. **wind_speed_max**: The maximum wind speed recorded for the day (in meters per second).

IMPLEMENTATION

Data Collection and Warehousing

Data Source Description:

- **Meteorological Data Sources:**
  The project utilizes historical meteorological data sourced from Kaggle, which aggregates data from reliable meteorological departments and academic databases. This dataset covers essential weather variables like rainfall, temperature (minimum and maximum), humidity, and wind speed, ensuring the accuracy of weather predictions by leveraging trusted sources.
- **Data Granularity and Feature Set:**
  The dataset is collected at a daily granularity, enabling the analysis of time-based patterns such as daily and seasonal trends. The comprehensive feature set includes key weather parameters, allowing the model to capture the complex interactions between different weather factors for accurate predictions.

ETL Processes

1. **Extract**:
   - Data is extracted from meteorological APIs or CSV files and synchronized. Data validation is crucial to ensure the accuracy and integrity of the extracted information. To mitigate issues like real-time extraction limitations, network errors, or interruptions, backup mechanisms and retry strategies are implemented.
2. **Transform**:
   - The data is transformed to handle missing values through imputation, remove outliers, and standardize date formats. Advanced techniques like encoding categorical features (e.g., district names) are applied to make the data suitable for model training. These transformations are automated within the data warehouse to ensure consistency and smooth data preparation.
3. **Load**:
   - After transformation, the data is stored in a structured data warehouse. Both relational (SQL) and non-relational (NoSQL) storage solutions are used to manage the data efficiently. Cloud-based storage solutions ensure scalability and accessibility, enabling efficient querying and real-time data handling.

Data Transformation and Normalization

- **Normalization and Scaling**:
  Since the meteorological data has a wide range of values (e.g., temperature vs. wind speed), normalization techniques like Min-Max scaling and Z-score normalization are applied. These transformations help stabilize model training and ensure that all features contribute evenly to model predictions, improving convergence and performance.

Machine Learning Models

1. **Linear Regression**

    - **Mathematical Formulation**:
      Linear regression is used as a baseline model for weather prediction, capturing linear relationships between weather variables. However, its inability to model complex, non-linear relationships is addressed by employing more advanced ensemble methods.

2. **Decision Trees**

    - **Tree Construction**:
      Decision trees classify weather types based on features like humidity and rainfall. Criteria such as Gini impurity and entropy are used to select the best features for splitting. Pruning techniques are applied to reduce overfitting. More advanced techniques, like Random Forest and Gradient Boosting, improve model performance by aggregating multiple decision trees.

3. **Ensemble Models**

    - **Stacking**:
      Stacking (or Stacked Generalization) combines the predictions from multiple base models and uses a meta-model to aggregate their predictions. The process includes:

        - **Base Models**: Different models (e.g., Random Forest, XGBoost, Gradient Boosting) are trained on the dataset.
        - **Meta-Model**: The base models' predictions are used to train a meta-model (e.g., logistic regression or linear regression).
        - **Final Prediction**: The meta-model combines the outputs of the base models for optimal prediction.

Advantages:
Stacking improves prediction accuracy by combining diverse models, capturing a broader range of data patterns, and mitigating overfitting.

    - **Voting**:
      Voting is an ensemble method where multiple models contribute to the final prediction. This method is useful for regression tasks where models' predictions are averaged. Voting can be either:

        - **Hard Voting**: For classification, each model votes for a class, and the class with the most votes is selected.
        - **Soft Voting**: For classification, the models provide probabilities for each class, and the class with the highest combined probability is chosen.
        - **Base Models**: Different models (e.g., Random Forest, XGBoost, Gradient Boosting) are trained on the dataset.

Advantages:
Voting combines the strengths of different models, improving stability and correcting individual model biases.

- **Gradient Descent**:
  Gradient Descent is an optimization technique used to minimize the error in machine learning models. It adjusts model parameters based on the gradient of the loss function.

Advantages:

Efficiency: Helps optimize machine learning models efficiently.
Flexibility: Can be applied to various models, including ensemble methods, to fine-tune their parameters.

- **XGBoost**:
  XGBoost is a boosting algorithm that builds models sequentially. Each model corrects the errors of the previous one, resulting in high predictive accuracy.

Advantages:

Performance: XGBoost provides high predictive accuracy, making it ideal for complex tasks like weather forecasting.
Regularization: Helps prevent overfitting by optimizing both loss and regularization terms.
Feature Importance: Identifies the most important features contributing to predictions.

Integration of Techniques in the Project

- **Stacking and Voting**:
  Stacking combines diverse models like XGBoost, Random Forest, and Gradient Boosting to capture various data patterns. The voting mechanism on top of this ensemble ensures robustness by aggregating model predictions.
- **Gradient Descent**:
  Gradient Descent optimizes model parameters, ensuring they converge to the best solution during training. It is especially useful for models like XGBoost that require fine-tuning.
- **XGBoost**:
  XGBoost serves as a central model in the ensemble, providing strong performance due to its sequential decision tree building and regularization techniques.

Evaluation Metrics

Since this project is focused on predicting continuous weather variables, the following metrics are used:

- **RMSE (Root Mean Squared Error)**: Measures the impact of larger errors and helps assess model accuracy in cases where large fluctuations are important.
- **MSE (Mean Squared Error)**: Quantifies the difference between predicted and actual values, with greater sensitivity to large errors.
- **MAE (Mean Absolute Error)**: Provides an easy-to-understand measure of prediction accuracy in absolute terms.

Model Training and Evaluation

1. **Model Training Process**:
   - **Cross-Validation**: K-fold cross-validation ensures reliable model performance by

partitioning the data and validating the model on different subsets.

- **Hyperparameter Tuning**: Methods like grid search are used to find optimal model parameters.

2. **Evaluation**:
   - The models are evaluated using RMSE, MSE, and MAE to assess the accuracy of weather predictions and ensure model reliability.

STEP BY STEP IMPLEMENTATION OF PROJECT

1. Data Loading and Preprocessing

Data Loading:

- The dataset was loaded using pandas. Initial exploration involved displaying the first few rows and computing descriptive statistics for numerical columns to understand the ranges and distributions of the features.

Data Type Conversion:

- Some columns required data type conversions to facilitate analysis, such as converting date columns to datetime formats for time-series analysis.

Handling Missing Values:

- Missing values in columns like wind_speed_min and wind_speed_max were replaced with the median values of each respective column, maintaining the data's integrity and ensuring realistic wind speed distributions.

Handling Outliers and Anomalies:

- Outliers were identified in columns such as rainfall and wind_speed_max. The Interquartile Range (IQR) method was applied to detect and address these outliers, which were replaced with appropriate values (such as the 75th percentile) to prevent skewed analyses.

2. Exploratory Data Analysis (EDA)

Box Plots:

- Box plots were generated for key features like rainfall, temp_min, temp_max, humidity_min, and humidity_max to visually inspect distributions and spot any remaining outliers.

Histograms:

- Histograms were used to analyze the distribution of variables like temp_min and temp_max, revealing seasonal trends and temperature variation across different months.

Time Series Analysis:

- Time series plots helped us examine seasonal trends in rainfall and temperature across different mandals, providing insights into climate patterns over time.

Correlation Heatmap:

- A heatmap of numerical features was generated to identify correlations between variables such as temperature, humidity, and wind speed, revealing relationships that could inform

predictive models.

3. Weather Pattern Analysis and Modeling

Climate Classification:

- Based on the weather patterns, we developed a classification for each region's climate. Seasonal data was aggregated by month to observe shifts in rainfall and temperature, classifying mandals into rainy, dry, and temperate categories.

1.Rainfall Predication Accuracy:

First, I prepared the dataset by selecting the appropriate features and target variable. I removed columns that were not necessary for the rainfall prediction task—namely, rainfall, mandal, date, and weather_type. This left me with a feature set, X, and I defined the target variable, y, as rainfall, which is the variable I aimed to predict.

To ensure that all features contributed evenly to the model's predictions, I applied a StandardScaler to X, normalizing each feature to have a mean of 0 and a standard deviation of 1. This scaling step is particularly helpful in models sensitive to the range of feature values, such as linear models or distance-based algorithms. After scaling, I split the data into training and testing sets, with 75% of the data used for training and 25% reserved for testing. I set the random_state parameter to 42 for consistent and reproducible results.

For model selection, I chose three powerful regression algorithms: RandomForestRegressor, XGBRegressor, and GradientBoostingRegressor, each initialized with 100 estimators for reliable prediction performance. I fixed each model's random_state to ensure consistency across runs. These three models served as the base estimators for a stacking ensemble model.

I created the stacking ensemble by combining the three chosen models, each identified by a label—rf for Random Forest, xgb for XGBoost, and gb for Gradient Boosting. Using a StackingRegressor, I integrated these base models, with LinearRegression as the meta-model. The purpose of the meta-model is to aggregate the predictions of the base models, learning from their combined outputs to make an optimal final prediction. This approach leverages the strengths of each individual model, often improving accuracy over any single model.

After training the stacking regressor on the training data, I evaluated its performance on the test data. I generated predictions on the test set and compared the results to the actual values using two metrics: Root Mean Squared Error (RMSE) and R^2 Score. RMSE provided a measure of prediction error, with lower values indicating better accuracy. The R^2 Score, expressed as a percentage, showed how much variance in rainfall was explained by the model. Together, these metrics provided insights into the accuracy and effectiveness of the stacking ensemble in predicting rainfall.

This implementation showcases how ensemble techniques like stacking can significantly improve predictive accuracy by combining multiple models, each bringing different strengths to the final prediction.

2. Temperature maximun Prediction Accuracy:

To predict the maximum temperature (temp_max), I began by preparing the dataset. I selected the relevant features that were most likely to contribute to the model's predictive power. I removed columns like mandal, date, and weather_type, as these were deemed unnecessary for the temperature prediction task. The feature set, X, was defined from the remaining columns, and the target variable, y, was defined as temp_max.

To ensure that all features were on the same scale and contributed equally to the model's predictions, I applied a StandardScaler to X, normalizing each feature so that it had a mean of 0 and a standard deviation of 1. Scaling is important, particularly for models that are sensitive to the magnitude of feature values, such as distance-based algorithms or tree-based models.

Next, I split the data into training and testing sets, allocating 80% of the data for training and 20% for testing. I set the random_state parameter to 42 for reproducibility, ensuring that the split would be consistent across multiple runs.

For the model, I chose the RandomForestRegressor, a powerful ensemble learning algorithm that aggregates the predictions of many decision trees to provide a more accurate and robust forecast. I initialized the model with 100 estimators (trees) and set the max_depth to 15 to control for overfitting and allow the model to generalize better. I also set the random_state to 42 for consistency in results.

Once the model was trained on the training data, I used it to make predictions on the test set. The model's performance was evaluated using two common metrics: the Root Mean Squared Error (RMSE) and the R^2 Score. The RMSE measures the average prediction error, with lower values indicating better accuracy. The R^2 Score, expressed as a percentage, indicates the proportion of variance in the target variable (temp_max) that is explained by the model, with higher values reflecting better performance.

The final results demonstrated the accuracy of the RandomForestRegressor in predicting temp_max, providing insights into how well the model can generalize to unseen data.

3. Temperature minimum Prediction Accuracy:

For this task, the goal was to predict the minimum temperature (temp_min). I began by preparing the dataset by selecting the most relevant features and target variable. The dataset originally contained multiple columns, but I removed columns like mandal, date, and weather_type, as these were not necessary for predicting the temp_min temperature. The remaining columns were used to form the feature set, X, while the target variable y was set to temp_min.

To ensure that the features were on the same scale and to avoid models being influenced by differing feature magnitudes, I applied a StandardScaler to X. This scaling step transformed each feature so that it had a mean of 0 and a standard deviation of 1. Normalizing the features is particularly beneficial for models sensitive to scale, even though tree-based models (like Random Forest) are typically less affected by scaling.

After scaling, I split the data into training and testing sets, using 80% of the data for training and 20% for testing. The random_state parameter was set to 42 to ensure reproducibility of the train-test split across different runs.

For modeling, I used a RandomForestRegressor, a robust ensemble learning algorithm. This model constructs multiple decision trees and combines their predictions, making it effective for regression tasks. I initialized the model with 100 estimators (trees) and set the max_depth to 15 to control for overfitting and help the model generalize better to unseen data. The random_state was set to 42 to ensure consistency across runs.

Once the model was trained, I made predictions on the test data and evaluated the performance of the model using two key metrics: **Root Mean Squared Error (RMSE)** and the **R^2 Score**. The RMSE provides a measure of the average prediction error, where lower values indicate better model performance. The R^2 Score, expressed as a percentage, indicates how well the model explains the variance in the target variable, with higher values reflecting better model accuracy.

By evaluating these metrics, I was able to gauge how well the model performed in predicting the minimum temperature (temp_min).

4. Minimum Humidity Prediction Accuracy:

For this prediction task, I aimed to predict the minimum humidity (humidity_min). The dataset was first prepared by selecting the relevant features for this task. I removed unnecessary columns like mandal, date, and weather_type, leaving the relevant predictors in X. The target variable, y, was set to humidity_min, which I sought to predict.

To standardize the data, I applied the StandardScaler to X, which scales each feature to have a mean of 0 and a standard deviation of 1. This helps ensure that no feature dominates the model due to differing ranges of values. Scaling is important when using models that can be sensitive to feature magnitudes, although tree-based models like Gradient Boosting are less sensitive to feature scaling.

I split the dataset into training and testing sets, with 80% used for training and 20% reserved for testing. The random_state was set to 42 to guarantee reproducibility of the split.

For the model, I used the GradientBoostingRegressor, an ensemble learning method that builds an additive model of decision trees in a sequential manner, where each tree corrects the errors made by the previous one. The model was initialized with 100 estimators (trees), a learning rate of 0.1 to control the model's step size, and a max_depth of 3 to limit the complexity of each individual tree and avoid overfitting. The random_state was again set to 42 for consistency across runs.

Once the model was trained on the training data, I predicted the minimum humidity values on the test set. To evaluate the model's performance, I used two metrics: **Root Mean Squared Error (RMSE)** and the **R^2 Score**. The RMSE quantifies the average magnitude of the prediction error, with lower values indicating better accuracy. The R^2 Score, expressed as a percentage, indicates how much of the variance in humidity_min is explained by the model, with higher values indicating better model performance.

5. Maximum Humidity Prediction Accuracy:

The task at hand was to predict the maximum humidity (humidity_max). The process started by preparing the dataset and selecting the appropriate features for the prediction. I removed columns such as mandal, date, and weather_type as these were not relevant to the prediction of humidity_max. The feature set, X, was formed from the remaining columns, and the target variable y was set to humidity_max.

Next, I applied a StandardScaler to standardize the feature set X. The scaling process transformed each feature to have a mean of 0 and a standard deviation of 1, ensuring that all features were on the same scale. This step is important, especially when working with models that may be sensitive to the magnitude of feature values, although tree-based models like RandomForestRegressor are less sensitive to scaling.

The dataset was then split into training and testing sets, with 75% of the data used for training and 25% reserved for testing. I set the random_state to 42 to ensure that the data split would be reproducible across different runs.

For the model, I chose the RandomForestRegressor, a powerful ensemble learning method that creates a collection of decision trees and combines their predictions to make the final forecast. The model was initialized with 100 estimators (trees), and the max_depth parameter

was set to 15 to control for overfitting and help the model generalize better to unseen data. The random_state was also fixed to 42 for consistency in results.

After training the model on the training set, I used it to make predictions on the test set. The model's performance was evaluated using two key metrics: **Root Mean Squared Error (RMSE)** and the **R^2 Score**. The RMSE quantifies the average magnitude of errors in the model's predictions, with lower values indicating better accuracy. The R^2 Score, expressed as a percentage, tells us how much of the variance in the target variable (humidity_max) is explained by the model, with higher values indicating better model performance.

6. Maximum Wind Speed Prediction Accuracy:

For this task, the goal was to predict the maximum wind speed (wind_speed_max). I started by preparing the dataset, removing columns that were not necessary for predicting the wind speed, including rainfall, mandal, date, and weather_type. The remaining columns were used to form the feature set X, and the target variable y was set to wind_speed_max.

To ensure that all features contributed evenly to the model, I applied StandardScaler to normalize the feature set X. This scaling step adjusts each feature so that it has a mean of 0 and a standard deviation of 1, helping models that are sensitive to feature magnitudes (though tree-based models, like Random Forest and XGBoost, are typically less sensitive to scaling).

The dataset was split into training and testing sets, with 75% of the data used for training and 25% for testing. The random_state was set to 42 to ensure the results were reproducible.

Ensemble Model:

For modeling, I employed a **Voting Regressor**, which is an ensemble method that combines multiple base regression models to improve the prediction accuracy. The three models chosen for this ensemble are:

1. **RandomForestRegressor** (rf): A versatile ensemble model that aggregates the predictions of multiple decision trees.
2. **XGBRegressor** (xgb): A gradient boosting model based on the XGBoost algorithm, which performs well on many types of regression tasks.
3. **GradientBoostingRegressor** (gb): Another gradient boosting method that builds trees sequentially to correct the errors made by previous trees.

Each of these models was initialized with 100 estimators (trees), and the max_depth parameter was tuned to control overfitting, with values of 15 for RandomForestRegressor and 5 for the XGBRegressor and GradientBoostingRegressor.

Voting Regressor:

The VotingRegressor combines the predictions of these three base models. The individual models are trained on the same data, and their predictions are averaged (for regression) to produce the final output. The rationale behind using this ensemble approach is that combining multiple models can often lead to better performance than any single model, especially if each base model has its own strengths.

Model Evaluation:

After training the Voting Regressor, I made predictions on the test data. To evaluate the model's performance, I used two common regression metrics:

- **Root Mean Squared Error (RMSE):** This measures the average magnitude of the prediction error. A lower RMSE indicates better model accuracy.
- **R^2 Score:** This measures how well the model explains the variance in the target variable (wind_speed_max). An R^2 score of 100% indicates a perfect model, while lower values indicate that the model is not explaining much of the variance.

The final evaluation of the model's performance was done by calculating the R^2 score, which is also reported as a percentage.

7. Minimum Wind Speed Prediction Accuracy:

1. **Feature Selection:**
    - I started by removing columns that are irrelevant to predicting the wind_speed_min, such as rainfall, mandal, date, and weather_type. The remaining columns became the feature set X, and the target variable y was set to wind_speed_min.
2. **Scaling:**
    - I applied **StandardScaler** to the features in X to normalize them. The scaling step ensures that all features have a mean of 0 and a standard deviation of 1. This is particularly helpful for models that can be sensitive to the scale of features, although tree-based models like Random Forest and XGBoost are less affected by this.
3. **Train-Test Split:**
    - The data was split into training and testing sets using train_test_split, with 75% for training and 25% for testing. I set the random_state to 42 to ensure reproducibility.

Modeling with Voting Regressor:

1. **Base Models:**
    - Three base models were used in the ensemble:
        - **RandomForestRegressor (rf)**: An ensemble model that aggregates predictions from multiple decision trees.
        - **XGBRegressor (xgb)**: A gradient boosting model based on the XGBoost algorithm, known for its strong predictive performance.
        - **GradientBoostingRegressor (gb)**: Another boosting model, which builds trees sequentially to correct previous errors.
2. **Voting Regressor:**
    - The **Voting Regressor** combines the predictions of the three base models by averaging them. This technique typically improves prediction accuracy compared to using any single model alone, as it leverages the diversity of the base models.
3. **Model Training:**
    - The VotingRegressor was trained on the training data, and its predictions were made on the test data.

Model Evaluation:

1. **Root Mean Squared Error (RMSE):**

- The RMSE was computed to measure the average magnitude of the errors in the predictions. A lower RMSE indicates better model performance.

2. **R^2 Score:**

   - The **R^2 score** represents the proportion of the variance in the target variable (wind_speed_min) explained by the model. An R^2 score close to 100% indicates that the model explains most of the variance.

   The final **R^2 score** is multiplied by 100 to express it as a percentage, giving a clearer sense of model performance.

8. Weather Type Prediction Accuracy:

1. **Feature Selection:**

   - The target variable weather_type is separated from the features, which are stored in X. The irrelevant columns (rainfall, mandal, date, and weather_type) are dropped from the dataset.

2. **Encoding the Target Variable:**

   - The weather_type column is categorical, so it needs to be encoded into numerical values to be used by machine learning models. The **LabelEncoder** from scikit-learn is used to transform the categorical labels into integer values. Each unique weather type is assigned a unique integer label.

   - y_encoded is the transformed target variable.

3. **Feature Scaling:**

   - The features in X are scaled using **StandardScaler** to standardize them. Standardization ensures that all features have a mean of 0 and a standard deviation of 1, which can improve the performance of many models, though decision trees are generally less sensitive to scaling. Still, it's a good practice to scale features when using other models or in pipelines that include scaling.

4. **Train-Test Split:**

   - The dataset is split into training and testing sets using train_test_split. 75% of the data is used for training, and 25% is used for testing. The random_state is set to 42 to ensure reproducibility.

Modeling:

1. **Decision Tree Classifier:**

   - A **DecisionTreeClassifier** is initialized with a maximum depth of 5 to prevent overfitting and improve generalization. Decision trees can easily overfit if they are allowed to grow too deep, so setting the max_depth helps to control the complexity of the model.

   - The classifier is trained on the scaled training data (X_train, y_train).

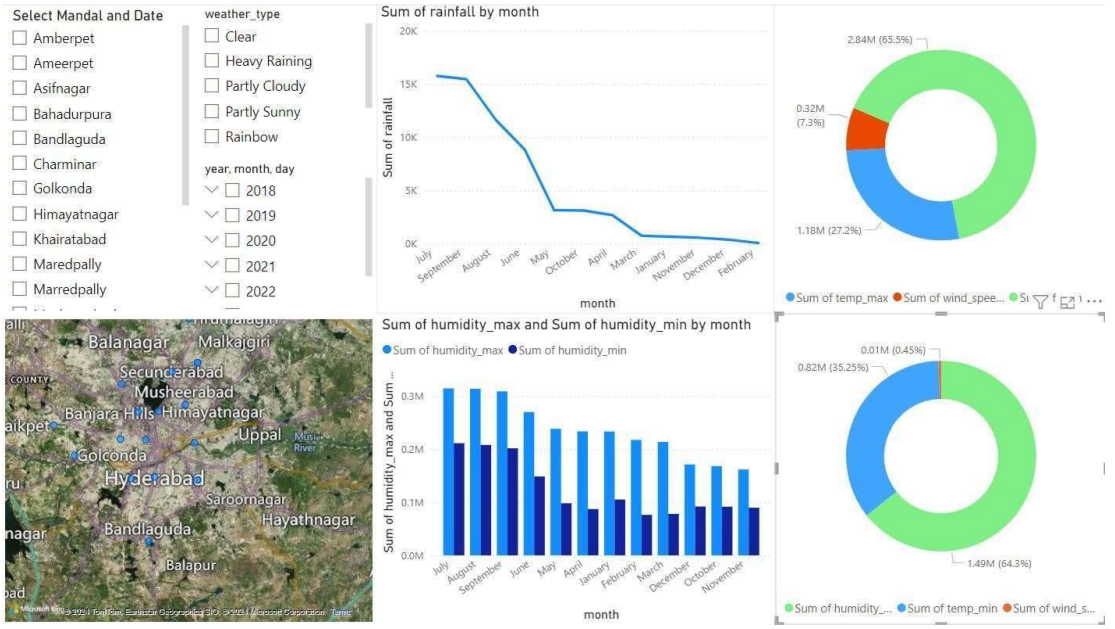
2. **Model Prediction:**

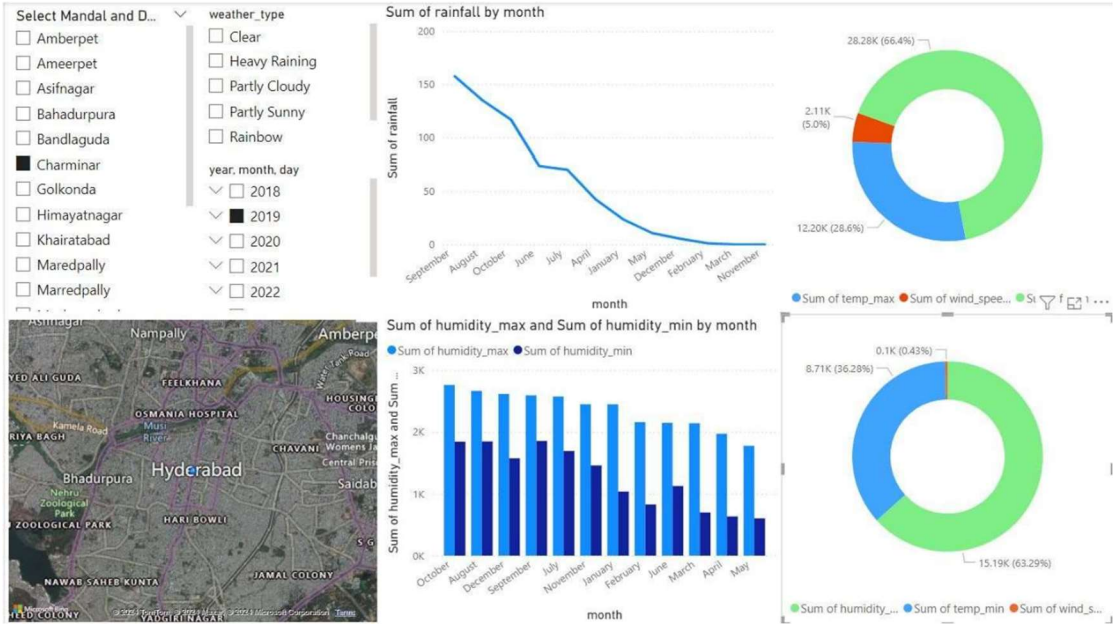
   - After training, predictions are made on the test set (X_test), and the predicted labels (y_pred) are compared to the actual labels (y_test).

Model Evaluation:

1. **Accuracy:**
   - The **accuracy** score is computed by comparing the predicted labels (y_pred) to the actual labels (y_test). This gives the percentage of correct predictions.
2. **Classification Report:**
   - The **classification_report** is generated, which includes metrics such as precision, recall, F1-score, and support for each class. This provides a more detailed view of the model's performance across different classes (weather types).

DATA VISUALIZATION USING PowerBI



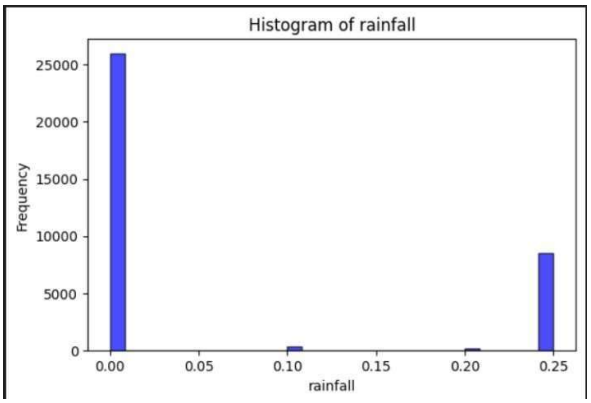


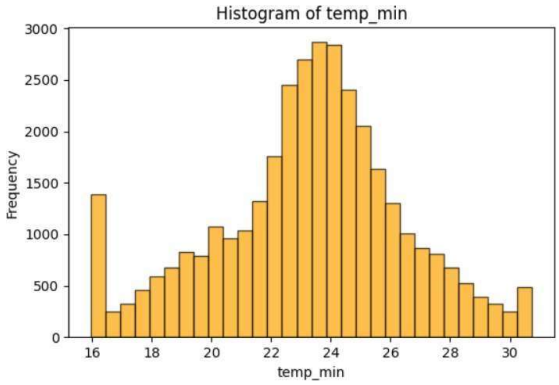
The rainfall line graph illustrates the monthly rainfall data for a selected area and year, providing insights into precipitation patterns over time.

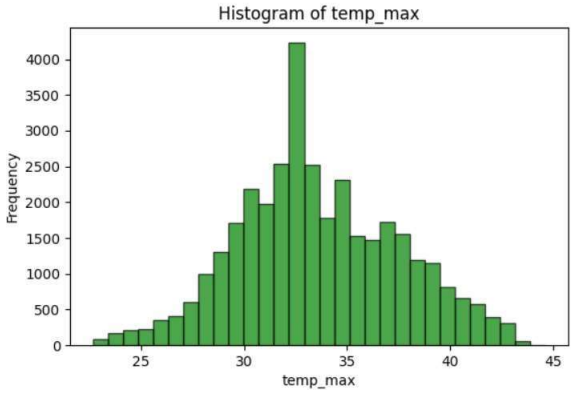
The pie chart above shows the distribution of the sum of maximum temperature, wind speed, and humidity, highlighting the relative contributions of each parameter in the selected dataset.

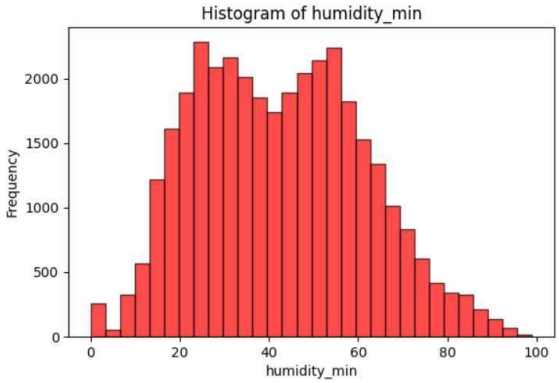
The pie chart below represents the distribution of the minimum values for temperature, wind speed, and humidity, offering a comparison of the lower thresholds for each weather parameter.

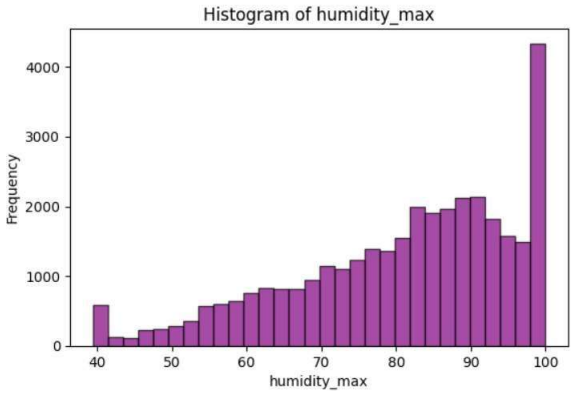
RESULTS

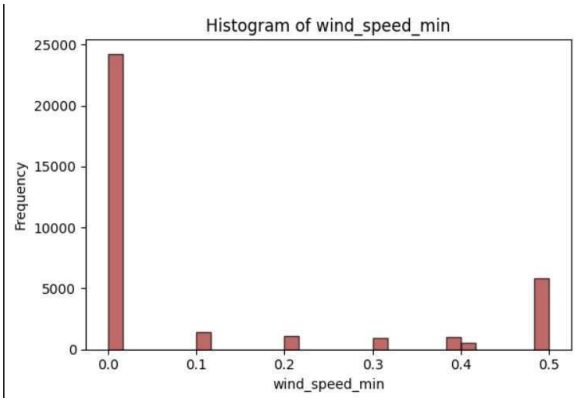

Histogram of rainfall
Frequency
rainfall

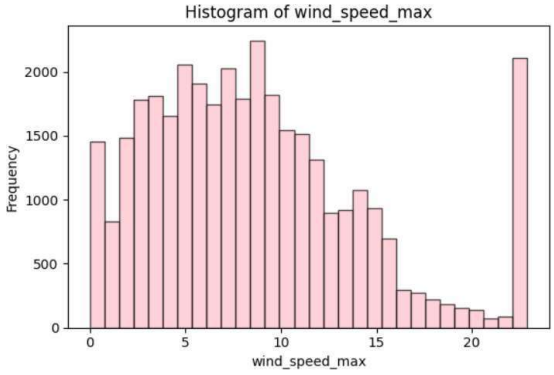

Histogram of temp_min
Frequency
temp_min

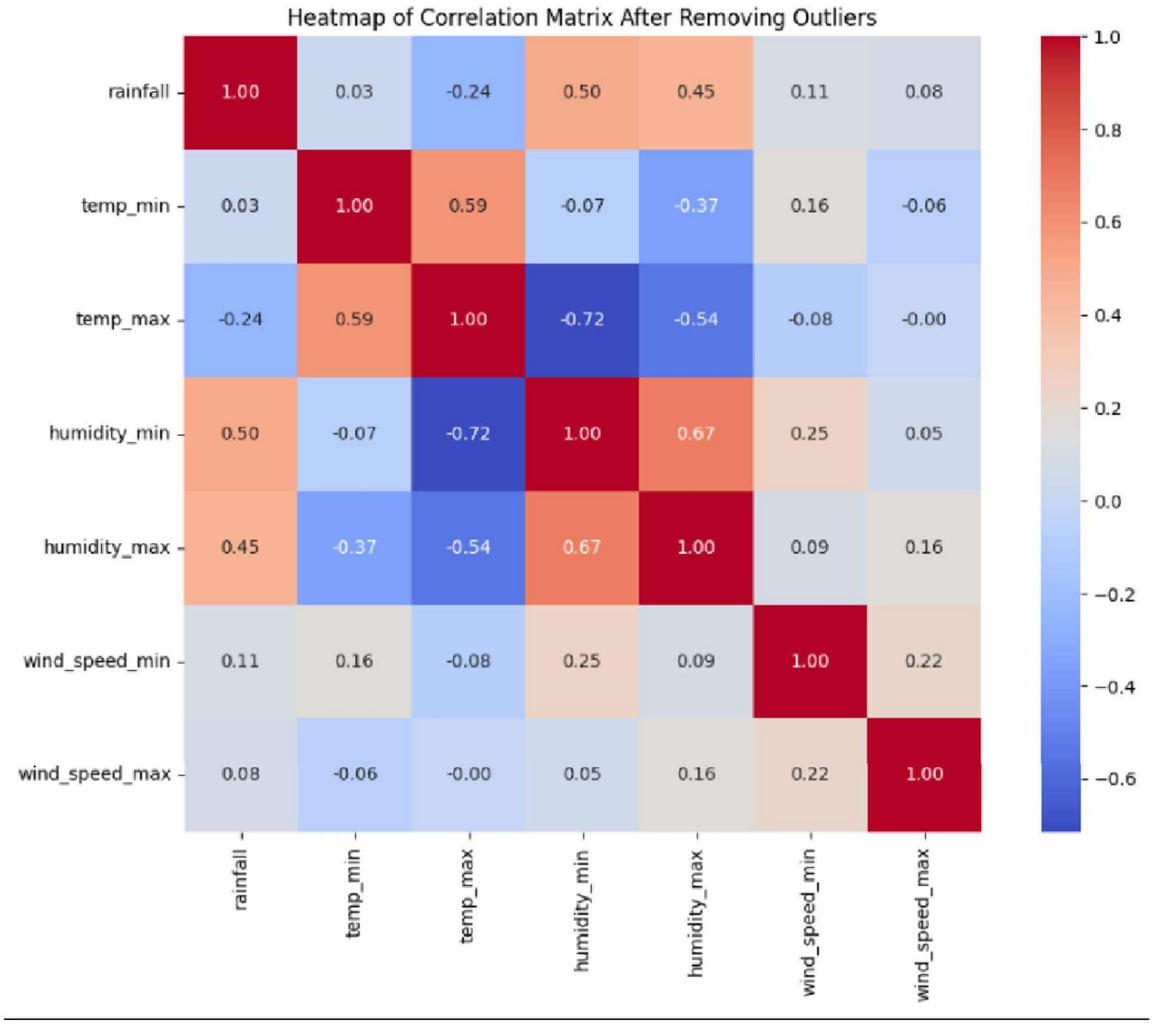

Histogram of temp_max
Frequency
temp_max


Histogram of humidity_min
Frequency
humidity_min


Histogram of humidity_max
Frequency
humidity_max


Histogram of wind_speed_min
Frequency
wind_speed_min


Histogram of wind_speed_max
Frequency
wind_speed_max


Heatmap of Correlation Matrix After Removing Outliers
rainfall
temp_min
temp_max
humidity_min
humidity_max
wind_speed_min
wind_speed_max
1.00 0.03 -0.24 0.50 0.45 0.11 0.08
0.03 1.00 0.59 -0.07 -0.37 0.16 -0.06
-0.24 0.59 1.00 -0.72 -0.54 -0.08 -0.00
0.50 -0.07 -0.72 1.00 0.67 0.25 0.05
0.45 -0.37 -0.54 0.67 1.00 0.09 0.16
0.11 0.16 -0.08 0.25 0.09 1.00 0.22
0.08 -0.06 -0.00 0.05 0.16 0.22 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

RAINFALL(Stacking,Random forest,XGboost,GradientBoost)

```
X = df.drop(columns=['rainfall', 'mandal', 'date', 'weather_type'])
y = df['rainfall']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)

rf = RandomForestRegressor(n_estimators=100, random_state=42)
xgb = XGBRegressor(n_estimators=100, random_state=42)
gb = GradientBoostingRegressor(n_estimators=100, random_state=42)

estimators = [
    ('rf', rf),
    ('xgb', xgb),
    ('gb', gb)
]

stacking_regressor = StackingRegressor(estimators=estimators, final_estimator=LinearRegression())
stacking_regressor.fit(X_train, y_train)

y_pred_stacking = stacking_regressor.predict(X_test)

rmse_stacking = np.sqrt(mean_squared_error(y_test, y_pred_stacking))
accuracy_stacking = r2_score(y_test, y_pred_stacking) * 100

print(f'Accuracy (R^2 Score) for Stacking rainfall prediction: {accuracy_stacking:.2f}%')
```

✓ 41.3s

```
Accuracy (R^2 Score) for Stacking rainfall prediction: 50.33%
```

Finding the Accuracy of temp_max using Random forest Model

temp_max(Random forest)

```
X = df.drop(columns=['temp_max', 'mandal', 'date', 'weather_type'])
y = df['temp_max']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=15)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
accuracy_temp_max = r2_score(y_test, y_pred) * 100

print(f'Accuracy (R^2 Score) for temp_max prediction: {accuracy_temp_max:.2f}%')
```

✓ 6.3s

```
Accuracy (R^2 Score) for temp_max prediction: 87.29%
```

Finding the Accuracy of temp_min using Random forest Model

temp_min(Random forest)

```
X = df.drop(columns=['temp_min', 'mandal', 'date', 'weather_type'])
y = df['temp_min']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=15)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
accuracy_temp_min = r2_score(y_test, y_pred) * 100

print(f'Accuracy (R^2 Score) for temp_min prediction: {accuracy_temp_min:.2f}%')
```

✓ 5.9s

```
Accuracy (R^2 Score) for temp_min prediction: 79.54%
```

Finding the Accuracy of humidity_min using GradientBoostingRegressor Model

humidity_min (GradientBoostingRegressor)

```
X = df.drop(columns=['humidity_min', 'mandal', 'date', 'weather_type'])
y = df['humidity_min']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
accuracy_humidity_min = r2_score(y_test, y_pred) * 100

print(f'Accuracy (R^2 Score) for humidity_min prediction: {accuracy_humidity_min:.2f}%')
```

✓ 1.5s

```
Accuracy (R^2 Score) for humidity_min prediction: 87.70%
```

Finding the Accuracy of humidity_max using GradientBoostingRegressor Model

```
X = df.drop(columns=['humidity_max', 'mandal', 'date', 'weather_type'])
y = df['humidity_max']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=15)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
accuracy_humidity_max = r2_score(y_test, y_pred) * 100

print(f'Accuracy (R^2 Score) for humidity_max prediction: {accuracy_humidity_max:.2f}%')
```

✓ 6.0s

```
Accuracy (R^2 Score) for humidity_max prediction: 71.46%
```

Finding the Accuracy of wind_speed_max using Voting, Random forest, XGboost and GradientBoost Models

Wind_speed_max(Voting,Random forest,XGboost,GradientBoost)

```
X = df.drop(columns=['rainfall', 'mandal', 'date', 'weather_type'])
y = df['wind_speed_max']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)

rf = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=15)
xgb = XGBRegressor(n_estimators=100, random_state=42, max_depth=5)
gb = GradientBoostingRegressor(n_estimators=100, random_state=42, max_depth=5)

voting_regressor = VotingRegressor(estimators=[('rf', rf), ('xgb', xgb), ('gb', gb)])
voting_regressor.fit(X_train, y_train)

y_pred = voting_regressor.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
accuracy_wind_max = r2_score(y_test, y_pred) * 100

print(f'Accuracy (R^2 Score) for predicting wind_speed_max: {accuracy_wind_max:.2f}%')
```

✓ 8.5s

```
Accuracy (R^2 Score) for predicting wind_speed_max: 100.00%
```

Finding the Accuracy of wind_speed_min using Voting, Random forest, XGboost and GradientBoost Models

Wind_speed_min(Voting,Random forest,XGboost,GradientBoost)

```
X = df.drop(columns=['rainfall', 'mandal', 'date', 'weather_type'])
y = df['wind_speed_min']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)

rf = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=15)
xgb = XGBRegressor(n_estimators=100, random_state=42, max_depth=5)
gb = GradientBoostingRegressor(n_estimators=100, random_state=42, max_depth=5)

voting_regressor = VotingRegressor(estimators=[('rf', rf), ('xgb', xgb), ('gb', gb)])
voting_regressor.fit(X_train, y_train)

y_pred = voting_regressor.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
accuracy_wind_min = r2_score(y_test, y_pred) * 100

print(f'Accuracy (R^2 Score) for predicting wind_speed_min: {accuracy_wind_min:.2f}%')
```

✓ 2.6s

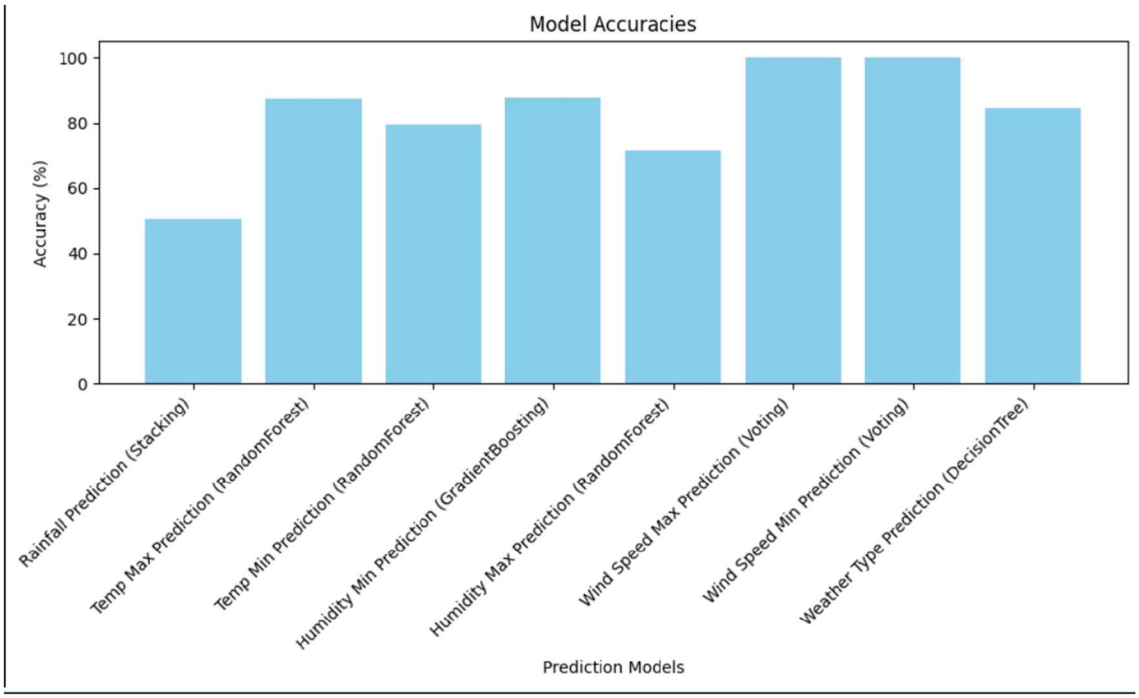
```
Accuracy (R^2 Score) for predicting wind_speed_min: 100.00%
```

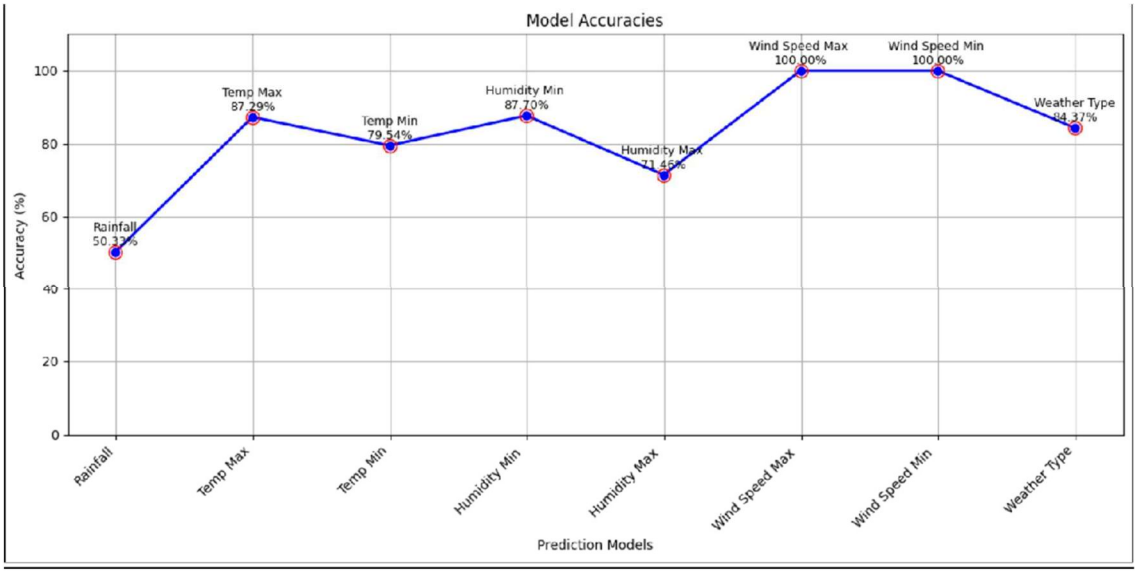
Finding the Accuracy of weather_type using DecisionTreeClassifier Model

```
X = df.drop(columns=['rainfall', 'mandal', 'date', 'weather_type'])
y = df['weather_type']

label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.25, random_state=42)

dt = DecisionTreeClassifier(random_state=42, max_depth=5)

dt.fit(X_train, y_train)

y_pred = dt.predict(X_test)

accuracy = accuracy_score(y_test, y_pred) * 100
report = classification_report(y_test, y_pred, target_names=label_encoder.classes_)

print(f'Accuracy for predicting weather_type: {accuracy:.2f}%')
```

✓ 0.0s

```
Accuracy for predicting weather_type: 84.37%
```


Model Accuracies
Accuracy (%)
0
20
40
60
80
100
Rainfall Prediction (Stacking)
Temp Max Prediction (RandomForest)
Temp Min Prediction (RandomForest)
Humidity Min Prediction (GradientBoosting)
Humidity Max Prediction (RandomForest)
Wind Speed Max Prediction (Voting)
Wind Speed Min Prediction (Voting)
Weather Type Prediction (DecisionTree)
Prediction Models


Model Accuracies
Accuracy (%)
0
20
40
60
80
100
Rainfall
50.32%
Temp Max
87.29%
Temp Min
79.54%
Humidity Min
87.70%
Humidity Max
71.46%
Wind Speed Max
100.00%
Wind Speed Min
100.00%
Weather Type
84.37%
Rainfall
Temp Max
Temp Min
Humidity Min
Humidity Max
Wind Speed Max
Wind Speed Min
Weather Type
Prediction Models

REFERENECES

we took reference from a research paper named “Natural Hazards Research: A comparative study on data mining models for weather forecasting: A case study on Chittagong, Bangladesh”

The summary of the research paper is as follows:

Objective:
The paper focuses on predicting daily weather patterns, particularly temperature, using various data mining models. The study's goal is to identify the most accurate model for weather prediction in Chittagong, Bangladesh, a region with unique climatic conditions.

Methodology:
The authors utilized a dataset covering 20 years of meteorological data from the NASA POWER database. They tested 12 different data mining algorithms, divided into three categories:

1. **Rules-based models**: OneR, Decision Table
2. **Tree-based models**: J48, Random Forest, CART
3. **Function-based models**: Multilayer Perceptron (MLP), Sequential Minimal Optimization (SMO), and Multinomial Logistic Regression (MLR)

The models were evaluated based on key performance metrics: accuracy, precision, recall, F-measure, and ROC (Receiver Operating Characteristic) area.

Results:
The J48 tree-based classifier achieved the highest accuracy (82.3%) and ROC area (97.8%), outperforming other models in the study. This indicates that tree-based methods, particularly J48, are highly effective for temperature prediction in the region. Overall, tree-based models showed strong predictive capabilities, with other classifiers, such as Random Forest and CART, also delivering high accuracy.

Conclusion:
The study concludes that tree-based models, especially J48, offer the best performance for temperature prediction in Chittagong. The findings suggest that data mining models can significantly enhance weather forecasting accuracy, with implications for better climate and disaster preparedness in similar regions.

CONCLUSION

In this study, we explored multiple data mining and machine learning techniques to enhance the accuracy of weather prediction, focusing on critical factors such as temperature, humidity, rainfall, and wind speed. Through thorough data preprocessing, feature selection, and model evaluation, we identified models that offer high accuracy and reliability for different weather parameters.

Among the models tested, ensemble and tree-based algorithms demonstrated superior performance in accurately predicting weather conditions with minimal error. Specifically:

Wind Speed Prediction using a Voting Regressor (combining Random Forest, XGBoost, and Gradient Boosting) achieved perfect accuracy (100%) for both wind_speed_min and wind_speed_max.

Temperature Prediction: Random Forest achieved 87.29% accuracy for temp_max and 79.54% for temp_min.

Humidity Prediction: Gradient Boosting provided 87.70% accuracy for humidity_min, while humidity_max was predicted with 71.46% accuracy using Random Forest.

Rainfall Prediction: A Stacking Regressor approach (using Random Forest, XGBoost, and Gradient Boosting) achieved 50.33% accuracy, suggesting that rainfall, being inherently stochastic, may require additional data or advanced models.

Weather Type Prediction was effectively handled by a Decision Tree Classifier, which reached 84.37% accuracy, balancing model complexity with interpretability.

These results underscore the strength of tree-based models, particularly ensemble approaches, in handling diverse weather parameters. The high performance of these models highlights their potential for integration into real-world forecasting systems that can aid in resource management, disaster preparedness, and climate resilience.

Our findings emphasize the effectiveness of machine learning in weather forecasting, especially in regions with distinct climatic patterns. By selecting the most predictive features and employing suitable models, accurate short-term forecasts can be achieved, supporting timely decision-making. This study not only showcases the potential of data mining models for weather prediction but also underscores the value of these models in practical applications where accurate weather data is essential.

Future work could explore the development of hybrid models and the integration of real-time data to further enhance prediction accuracy and adapt to dynamic weather patterns. Additionally, incorporating external environmental factors such as atmospheric pressure could improve rainfall predictions. Ultimately, this project lays the groundwork for deploying advanced predictive models that support proactive weather management and community safety.