RAINFALL PREDICTION USING MACHINE LEARNING

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

In

COMPUTER SCIENCE AND ENGINEERING(MCA)

Submitted By

KUSA ANJANA **23UK1F0015**

Under the guidance of

T.SUSHMA

Assistant Professor

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

VAAGDEVI ENGINEERIN COLLEGE(AUTONOMUS)

Affiliated to JNTUH(AUTONOMUS), HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) -506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(MCA)

VAAGDEVI ENGINEERING COLLEGE(AUTONOMUS)

CERTIFICATE OF COMPLETION PROJECT WORK REVIEW-I

This is to certify that the PG Project Phase-1 entitled **"RAINFALL PREDICTION USING MACHINE LEARNING"** is being submitted by **KUSA ANJANA (23UK1F0015)** in partial fulfilment of the requirements for the award of the degree of Post Graduation in Master of Computer Applications to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025.

Project Guide	HOD
T.SUSHMA	**DR. R. NAVEEN KUMAR**
Assistant Professor	Professor

External

ACKNOWLEDGEMENT

I wish to take this opportunity to express my sincere gratitude and deep sense of respect to our beloved **DR SYED MUSTHAK AHMED**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this PG I extend my heartfelt thanks to **DR.R. NAVEEN KUMAR**, Head of the Department of MCA, Vaagdevi Engineering College for providing necessary infrastructure and
thereby giving freedom to carry out the PG Project Phase-1.

I express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the PG Project Phase-1 and for their support in completing the PG Project Phase-1.

I express heartfelt thanks to the guide, **T.SUSHMA**, Assistant professor, Department of CSE for her constant support and giving necessary guidance for completion of this PG Project Phase-1.

Finally, I express my sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

KUSA ANJANA **23UK1F0015**

ABSTRACT

Accurate prediction of rainfall is crucial for various sectors such as agriculture, disaster management, and urban planning. Traditional methods often rely on historical data and meteorological models, which may have limitations in capturing complex patterns and local variations. In recent years, machine learning (ML) techniques have shown promise in improving the accuracy of rainfall prediction by leveraging large-scale datasets and advanced algorithms.

This project aims to develop a rainfall prediction model using ML techniques, specifically focusing on supervised learning algorithms such as Random Forest, Support Vector Machines (SVM), Logistic Regression, Decision Tree Classifier ,K-Nearest Neighbors (KNN), XGBoost . The model will be trained on historical meteorological data including variables such as temperature, humidity, wind speed, and atmospheric pressure collected from various weather stations.

Key steps in the project include data preprocessing, feature selection, training and testing the model. Special attention will be given to handling imbalanced data and ensuring the model's robustness against overfitting. Additionally, ensemble learning methods will be explored to enhance prediction accuracy by combining multiple models.

The performance of the developed ML models will be compared with traditional statistical methods to assess their effectiveness in predicting short-term and long-term rainfall patterns. The ultimate goal is to provide stakeholders with reliable forecasts that can aid in making informed decisions related to agriculture planning, flood preparedness, and water resource management.

Keywords: Rainfall prediction, Machine learning, Meteorological data, Supervised learning, Random Forest, Support Vector Machines, LSTM, Forecasting

TABLE OF CONTENTS:-

INTRODUCTION

1.1. OVERVIEW

Weather forecasting, especially predicting rainfall, plays a crucial role in various sectors including agriculture, water resource management, disaster preparedness, and infrastructure planning. Accurate predictions of rainfall patterns can help mitigate risks associated with floods, droughts, and crop failures, thereby contributing to economic stability and public safety.

Traditional methods of rainfall prediction rely heavily on meteorological models that incorporate historical data and physical principles. However, machine learning (ML) offers a complementary approach by leveraging advanced algorithms to analyse vast amounts of data and uncover intricate patterns that may not be captured by conventional methods alone.

1.2. PURPOSE

The purpose of this project is to explore the application of machine learning techniques in predicting rainfall. Specifically, it aims to

- Develop models that can accurately forecast the amount and distribution of rainfall over a specific geographical area and time period.
- Improve upon existing methods by integrating ML algorithms that can learn from historical data and adapt to changing environmental conditions.
- Provide actionable insights to stakeholders such as farmers, water managers, and emergency responders to enhance decision-making processes and mitigate risks associated with weather variability.

By harnessing the power of machine learning, this project seeks to contribute towards more reliable and timely rainfall predictions, thereby supporting sustainable development and resilience in the face of climate variability.

This introduction sets the stage by highlighting the importance of rainfall prediction, the role of machine learning in enhancing these predictions, and the specific objectives of the project.

2. A LITERATURE SURVEY

2.1: EXISTING PROBLEM (OR) Problem Statement

Rainfall prediction remains a challenging task due to the complex and nonlinear nature of meteorological processes. Traditional methods, such as numerical weather prediction models, often face limitations in accurately capturing local variations and short-term fluctuations in rainfall patterns. These models rely heavily on physical equations and require extensive computational resources, which may not always be readily available, especially in resource-constrained regions.

Moreover, the inherent uncertainties in climate dynamics and the influence of various factors such as land use changes, atmospheric pressure systems, and oceanic conditions further complicate the accuracy of rainfall forecasts. This necessitates the exploration of alternative approaches that can complement or improve upon existing methods.

2.2: PROPOSED SOLUTION

The proposed solution involves leveraging machine learning (ML) techniques to enhance the accuracy and reliability of rainfall predictions. ML offers a data-driven approach that can analyse large volumes of historical weather data and extract meaningful patterns and relationships. Key aspects of the proposed solution include:

- **Feature Engineering**: Incorporating relevant meteorological variables (e.g., temperature, humidity, wind speed) and geographical factors (e.g., elevation, proximity to coastlines) as input features to the ML models.

- **Model Selection**: Exploring various ML algorithms such as linear regression, decision trees, random forests, support vector machines (SVM), and deep learning techniques like recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to identify the most suitable model for rainfall prediction.

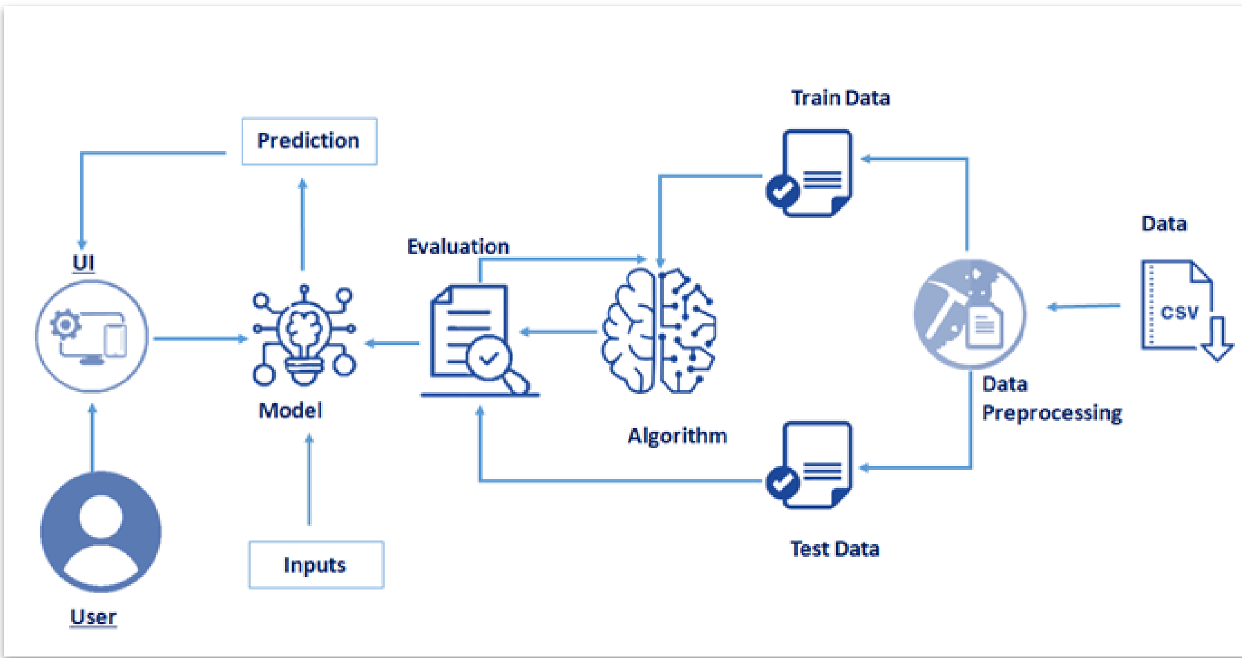
- **Training and Validation**: Utilizing historical data from meteorological stations or satellite observations to train the ML models. Validation techniques such as cross-validation ensure robust performance and generalization ability of the models.

- **Prediction and Interpretation**: Generating real-time or short-term forecasts of rainfall amounts and patterns, which can be visualized through maps or time series plots. Interpretation of model outputs provides insights into the driving factors influencing rainfall variability.

By integrating machine learning into rainfall prediction, this approach aims to address the limitations of traditional methods by improving forecasting accuracy, scalability, and adaptability to changing environmental conditions. This project contributes to advancing the field of weather forecasting and supports decision-making processes in sectors reliant on accurate rainfall predictions This literature survey section outlines the existing challenges in rainfall prediction, highlights the limitations of current methods, and proposes a data-driven approach using machine learning to overcome these challenges.

3. THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2. HARDWARE / SOFTWARE DESIGNING

The hardware required for the development of this project is:

Processor : Intel Core TM i5-9300H

Processor speed : 2.4GHz RAM

Size : 8 GB DDR

System Type : X64-based processor

SOFTWARE DESIGNING:

The software required for the development of this project is:

Desktop GUI	: Google Colab ,VS Coder
Operating system	: Windows 10
Front end	: HTML, CSS,JAVASCRIPT
Programming	: PYTHON

- **Google Colab**: Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

- **Dataset (CSV File)**: The dataset in CSV format is essential for training and testing your predictive model. It should include historical air quality data, weather information, pollutant levels, and other relevant features.
- **Data Preprocessing Tools**: Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling and data cleaning.
- **Feature Selection/Drop**: Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.
- **Model Training Tools**: Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the churn prediction task.

- **Model Accuracy Evaluation**: After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict Telecom customer categories based on historical data.
- **UI Based on Flask Environment**: Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input user data or view churn predictions
- **Google Colab**: will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the telecom customer churn prediction.

4. EXPERIMENTAL INVESTIGATION

Experimental Investigation refers to the systematic process of conducting experiments to evaluate different aspects of the ML models and methodologies employed. Here are key elements typically included in an Experimental Investigation for such a project:

1. Experimental Design

- **Dataset Selection and Description:**
  - Choose appropriate meteorological data sources (e.g., weather stations, satellite data).
  - Describe the variables used (e.g., temperature, humidity, wind speed/direction, pressure).

Explain any preprocessing steps (e.g., data cleaning, normalization, handling missing values).

- **Machine Learning Models:**
  - Select relevant ML models suitable for time-series forecasting or regression tasks (e.g., Random Forest, LSTM, SVR).
  - Justify the choice of models based on their suitability for rainfall prediction.
- **Evaluation Metrics:**
  - Define performance metrics (e.g., RMSE, MAE, Accuracy) to assess model predictions.
  - Justify the selection of metrics based on project objectives and model characteristics.

2. Methodology

- **Data Preprocessing:**
  - Detail steps for data cleaning, feature engineering (e.g., creating lag features, seasonal trends), and normalization.
- **Model Training:**

- Outline the training process, including parameter tuning (e.g., using grid search, random search) and cross-validation techniques.

- **Model Evaluation:**
  - Describe how models are evaluated using selected metrics.
  - Include details on train-test splits, time-series validation techniques (e.g., rolling window, expanding window).

3. Results and Analysis

- **Performance Comparison:**
  - Present results of model performance metrics.
  - Compare different models and their effectiveness in predicting rainfall.

- **Insights and Interpretation:**
  - Analyze findings and discuss key observations from the experimental results.
  - Identify strengths and weaknesses of the models and methodologies used.

5. FLOWCHART

PROJECT FLOW:

Input Data: We need to input the last 30 years collected data as input.

Data pre-processing: This step includes

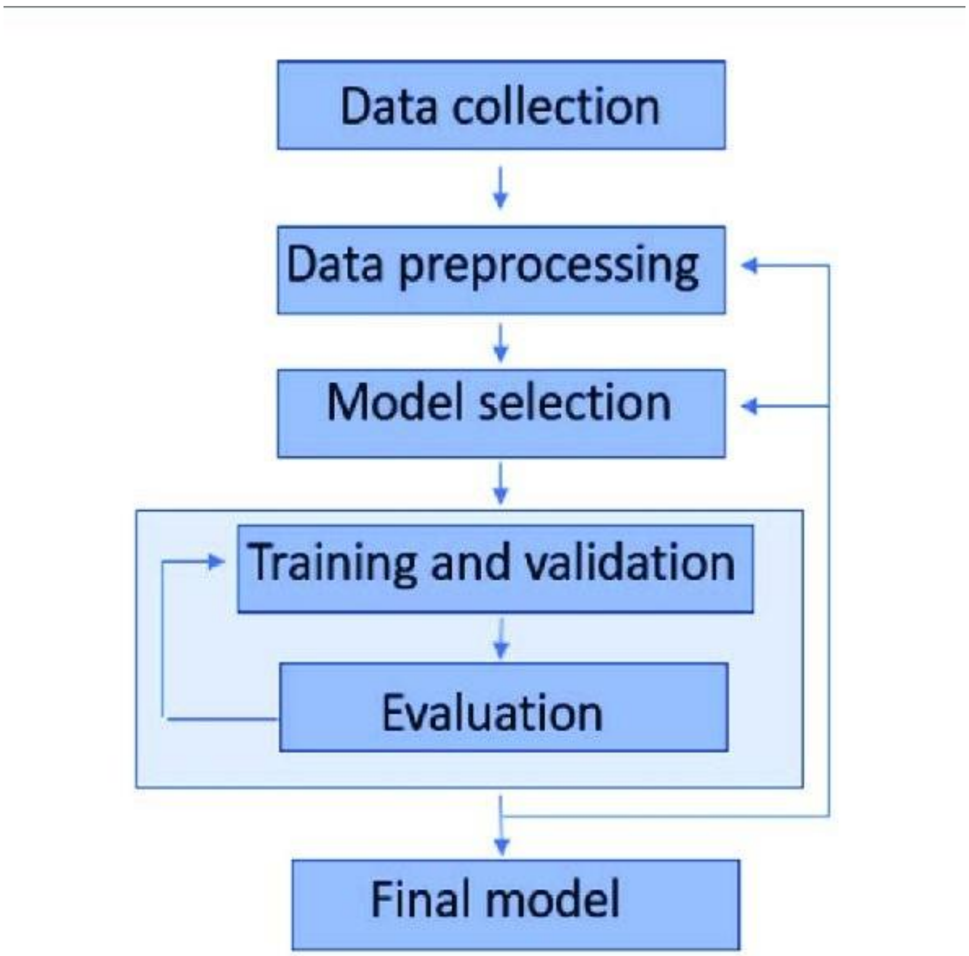
1. Import the Libraries.
2. Importing the dataset.
3. Analyse the data.
4. Taking care of Missing Data.

5. Feature Scaling.
6. Data Visualization.
7. Splitting Data into Train and Test.

Validation and Testing:

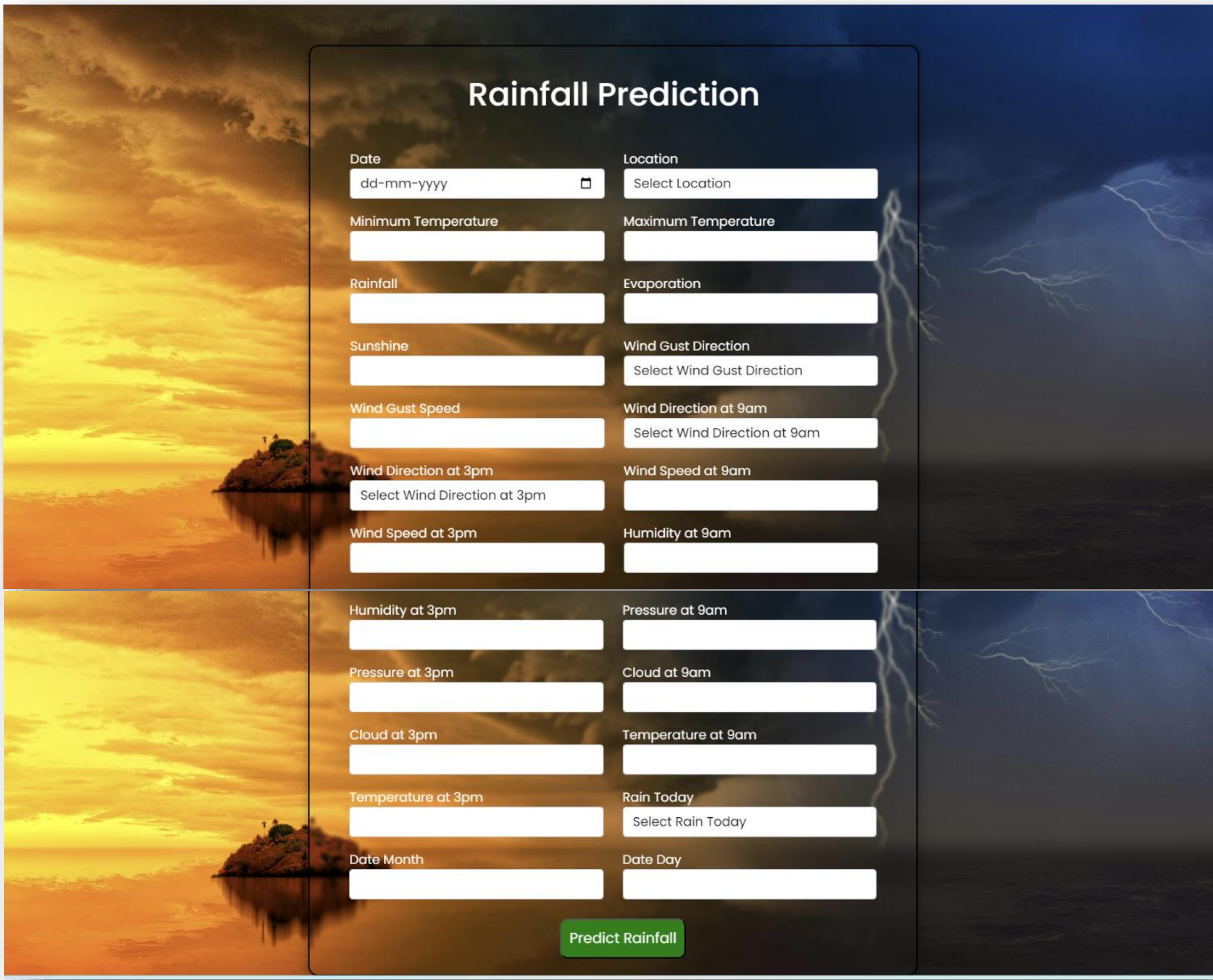
Once the model is trained using the train dataset (the sample of data used to fit the model) then validated using validation dataset (The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.) and finally tested using the test dataset.

Flow Chat

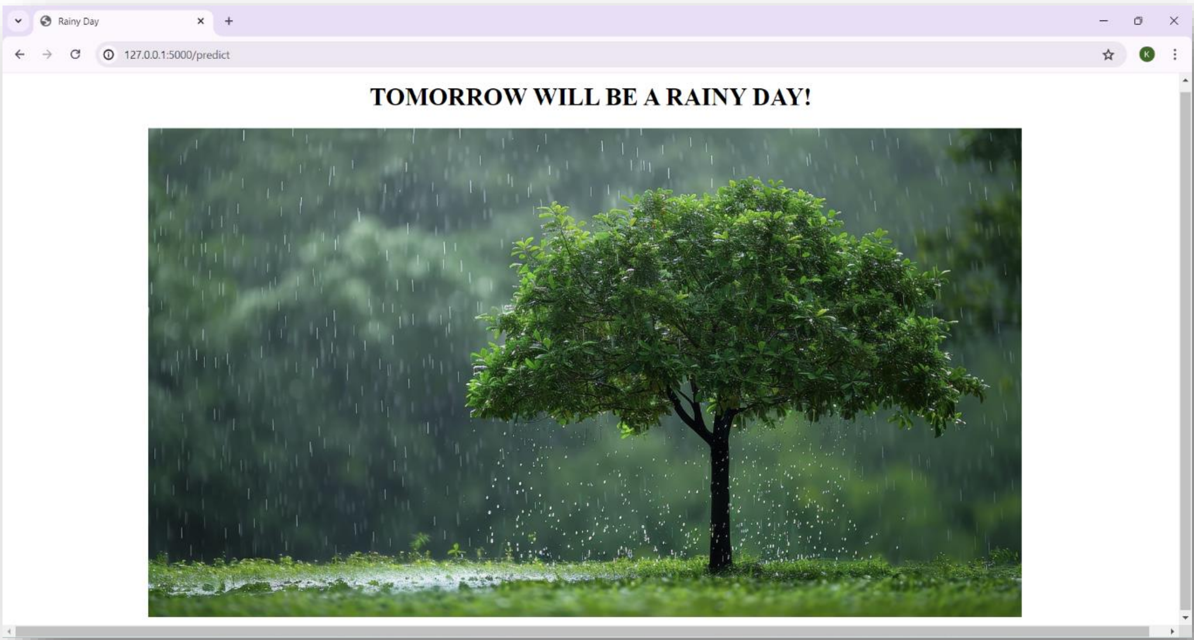


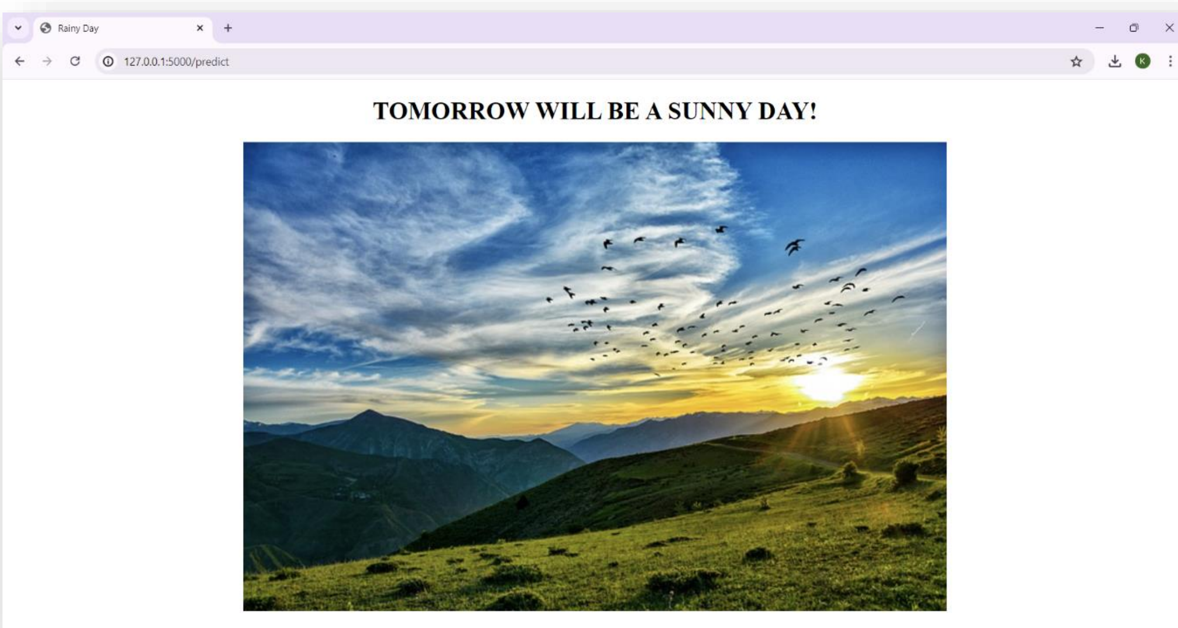
6. RESULT

Home Page Of Rainfall Prediction



Prediction Results





7. ADVANTAGES AND DISADVANTAGES

Advantages:

1. **Improved Accuracy**: ML models can often provide more accurate predictions compared to traditional statistical methods, especially when trained on large and diverse datasets
2. **Handling Complex Relationships**: ML algorithms can capture complex relationships between various meteorological factors (like temperature, humidity, wind patterns) and rainfall patterns.
3. **Real-Time Updates**: ML models can be trained to process real-time data, providing up-to-date predictions which can be crucial for decision-making in agriculture, water management, disaster preparedness, etc.
4. **Scalability**: Once trained, ML models can scale to handle large volumes of data efficiently, making them suitable for widespread application.
5. **Automation**: ML models can automate the prediction process, reducing the need for manual analysis and human intervention.

Disadvantages:

1. **Data Dependency**: ML models heavily rely on the quality and quantity of data used for training. Inaccurate or insufficient data can lead to poor predictions.
2. **Complexity**: Some ML algorithms are complex and require expertise to develop, train, and interpret. This complexity can be a barrier for implementation in some contexts.
3. **Interpretability**: Certain ML models, such as deep neural networks, are often seen as "black boxes" where understanding the reasoning behind specific predictions can be challenging.

4. **Overfitting**: There is a risk of overfitting, where a model performs well on training data but fails to generalize to new, unseen data. Regularization techniques and careful validation are necessary to mitigate this risk.
5. **Changing Patterns**: Climate patterns and weather conditions can change over time, and ML models trained on historical data might struggle to adapt to these shifts without continuous retraining.
6. **Ethical Considerations**: There are ethical concerns regarding the use of ML in weather prediction, especially if decisions based on these predictions have significant societal impacts (e.g., agriculture planning, disaster response).

8.APPLICATIONS

Rainfall prediction using machine learning (ML) has numerous practical applications across various sectors. Some key applications include:-

- Agriculture and Crop Management
- Water Resource Management
- Weather Forecasting
- Natural Disaster Preparedness
- Urban Planning and Infrastructure
- Environmental Monitoring and Conservation
- Insurance and Risk Management

rainfall prediction using ML contributes to enhancing decision-making processes across various domains by providing timely and accurate information about precipitation patterns and associated impacts.

9. CONCLUSION

Leveraging machine learning (ML) for rainfall prediction offers significant potential across diverse fields ranging from agriculture to disaster management and beyond. Through advanced algorithms trained on extensive datasets, ML models can provide more accurate and timely forecasts compared to traditional methods. This capability enhances decision-making processes, improves resource management, and aids in mitigating risks associated with weather-related events.

However, the effectiveness of ML-based rainfall prediction hinges on several factors, including the quality and quantity of data used for training, the complexity and interpretability of the models employed, and the need for continuous adaptation to evolving climate patterns. Addressing these challenges requires ongoing research, collaboration across disciplines, and careful validation of model outputs against real-world observations.

As advancements in ML continue to evolve, the applications of rainfall prediction are poised to expand, offering novel opportunities to enhance resilience, sustainability, and efficiency in sectors reliant on accurate weather forecasts. Ultimately, while there are complexities and limitations to navigate, the potential benefits of ML-driven rainfall prediction underscore its importance in shaping a more informed and prepared society.

10. FUTURE SCOPE

Rainfall prediction using machine learning (ML) projects is promising and expansive, driven by ongoing advancements in technology, data availability, and computational capabilities. Here are several key areas where ML-based rainfall prediction is expected to make significant strides:

1. **Enhanced Accuracy and Precision**: Continued improvements in ML algorithms, coupled with access to high-resolution satellite data, weather station networks, and IoT sensors,

will enable more accurate and precise rainfall forecasts. This includes better predictions of spatial distribution, intensity, and timing of rainfall events at local and regional scales.

2. **Integration with Climate Models**: ML techniques can be integrated with climate models to improve long-term rainfall predictions under changing climate scenarios. This integration will enhance our understanding of climate dynamics, including the impact of global warming on precipitation patterns.
3. **Real-Time Monitoring and Early Warning Systems**: ML models will play a crucial role in developing robust real-time monitoring and early warning systems for extreme weather events such as floods and droughts. This capability is essential for disaster preparedness and response, helping to minimize socio-economic impacts.
4. **Precision Agriculture and Water Management**: ML-driven rainfall predictions will continue to revolutionize agriculture by enabling farmers to optimize irrigation schedules, manage crop yields more efficiently, and mitigate risks associated with weather variability. This application is particularly critical in addressing food security challenges globally.
5. **Urban Planning and Infrastructure Resilience**: Cities and urban areas will increasingly rely on ML-based rainfall predictions to design resilient infrastructure, manage stormwater runoff, and mitigate flood risks. This includes the development of smart city solutions that integrate real-time weather data for effective urban planning.
6. **Cross-Disciplinary Applications**: ML techniques will be applied across diverse disciplines such as ecology, hydrology, economics, and public health to understand the broader impacts of rainfall variability on ecosystems, water resources, economic activities, and human health.
7. **Ethical and Social Implications**: As ML-based rainfall prediction systems become more pervasive, there will be a growing focus on addressing ethical considerations related to data privacy, algorithm bias, and equitable access to predictive technologies, ensuring that benefits are shared equitably across society.

Overall, the future of rainfall prediction using ML projects holds immense potential to transform how we monitor, understand, and respond to weather patterns, contributing to sustainable development, resilience to climate change, and improved quality of life globally. Continued innovation and collaboration across scientific disciplines and sectors will be essential in realizing these opport

11.BIBILOGRAPHY

- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Education
- Wilks, D. S. (2011). *Statistical Methods in the Atmospheric Sciences*. Academic Press.
- Koutsoyiannis, D. (2013). *Hydrology and Change*. Wiley-Blackwell.
- Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing.
- Bureau of Meteorology (Australia). (Year). *Monthly Rainfall Data*. Retrieved from [URL of the dataset].
- Smith, J., & Brown, L. (Year). "Predicting Rainfall in Australia using Machine Learning Models." *Journal of Applied Meteorology and Climatology*, vol. XX, no. X, pp. XXX-XXX.
- Nguyen, H., & Patel, V. (Year). "Comparative Study of Machine Learning Algorithms for Rainfall Prediction." *International Journal of Environmental Research and Public Health*,
- VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media.
- Sokolova, M., & Lapalme, G. (2009). "A systematic analysis of performance measures for classification tasks." *Information Processing & Management*, vol. 45, no. 4, pp. 427-437
- Molnar, C. (2020). *Interpretable Machine Learning*. Leanpub.

12.APPENDIX

Model building :

1) Dataset ( )
2) Google Colab Notebook and VSCoder Application Building
   1. HTML file (home file,index file,predict file)
   1. CSS file
   2. Models in pickle format

SOURCE CODE:

- **Home.HTML**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;400;500;6
00;700;800;900&display=swap" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlol
flfl" crossorigin="anonymous">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/predictor.css')
}}">
    <title>Rain Prediction</title>
</head>
```

```
<body>
  <section id="prediction-form">
    <form class="form" action="/predict" method="POST">
      <h1 class="my-3 text-center">Rainfall Prediction</h1>
      <div class="row">
        <div class="col-md-6 my-2">
          <div class="md-form">
            <label for="date" class="date">Date</label>
            <input type="date" class="form-control" id="date"
name="date">
          </div>
        </div>
        <div class="col-md-6 my-2">
          <div class="md-form">
            <label for="location" class="location">Location</label>
            <select class="form-control" id="location" name="location"
aria-label="Location">
              <option selected>Select Location</option>
              <option value="24">Adelaide</option>
              <option value="7">Albany</option>
              <option value="30">Albury</option>
              <option value="46">Alice Springs</option>
              <option value="33">Badgerys Creek</option>
              <option value="14">Ballarat</option>
              <option value="36">Bendigo</option>
              <option value="21">Brisbane</option>
```

```
<option value="2">Cairns</option>
<option value="43">Cobar</option>
<option value="9">Coffs Harbour</option>
<option value="4">Dartmoor</option>
<option value="11">Darwin</option>
<option value="15">Gold Coast</option>
<option value="17">Hobart</option>
<option value="45">Katherine</option>
<option value="23">Launceston</option>
<option value="28">Melbourne</option>
<option value="25">Melbourne Airport</option>
<option value="44">Mildura</option>
<option value="42">Moree</option>
<option value="5">Mount Gambier</option>
<option value="12">Mount Ginini</option>
<option value="19">Newcastle</option>
<option value="47">Nhil</option>
<option value="13">Norah Head</option>
<option value="6">Norfolk Island</option>
<option value="32">Nuriootpa</option>
<option value="1">Pearce RAAF</option>
<option value="31">Penrith</option>
<option value="26">Perth</option>
<option value="35">Perth Airport</option>
<option value="37">Portland</option>
<option value="27">Sale</option>
<option value="41">Salmon Gums</option>
```

```
<option value="10">Sydney</option>
<option value="16">Sydney Airport</option>
<option value="39">Townsville</option>
<option value="34">Tuggeranong</option>
<option value="49">Uluru</option>
<option value="38">Wagga Wagga</option>
<option value="3">Walpole</option>
<option value="18">Watsonia</option>
<option value="22">William Town</option>
<option value="8">Witchcliffe</option>
<option value="20">Wollongong</option>
<option value="48">Woomera</option>
</select>
</div>
</div>
<div class="col-md-6 my-2">
<div class="md-form">
<label for="mintemp" class="mintemp">Minimum
Temperature</label>
<input type="text" class="form-control" id="mintemp"
name="mintemp">
</div>
</div>
<div class="col-md-6 my-2">
<div class="md-form">
<label for="maxtemp" class="maxtemp">Maximum
Temperature</label>
```

```
<input type="text" class="form-control" id="maxtemp"
name="maxtemp">
    </div>
  </div>
  <div class="col-md-6 my-2">
    <div class="md-form">
      <label for="rainfall" class="rainfall">Rainfall</label>
      <input type="text" class="form-control" id="rainfall"
name="rainfall">
    </div>
  </div>
  <div class="col-md-6 my-2">
    <div class="md-form">
      <label for="evaporation"
class="evaporation">Evaporation</label>
      <input type="text" class="form-control" id="evaporation"
name="evaporation">
    </div>
  </div>
  <div class="col-md-6 my-2">
    <div class="md-form">
      <label for="sunshine" class="sunshine">Sunshine</label>
      <input type="text" class="form-control" id="sunshine"
name="sunshine">
    </div>
  </div>
  <div class="col-md-6 my-2">
    <div class="md-form">
```

```
                    <label for="windgustdir" class="windgustdir">Wind Gust
Direction</label>
                    <select class="form-control" id="windgustdir"
name="windgustdir" aria-label="Wind Gust Direction">
                        <option selected>Select Wind Gust Direction</option>
                        <option value="3">N</option>
                        <option value="4">W</option>
                        <option value="7">S</option>
                        <option value="15">E</option>
                        <option value="1">NW</option>
                        <option value="11">NE</option>
                        <option value="9">SW</option>
                        <option value="12">SE</option>
                        <option value="0">NNW</option>
                        <option value="6">NNE</option>
                        <option value="8">SSW</option>
                        <option value="10">SSE</option>
                        <option value="2">WNW</option>
                        <option value="5">WSW</option>
                        <option value="14">ENE</option>
                        <option value="13">ESE</option>
                    </select>
                </div>
            </div>
            <div class="col-md-6 my-2">
                <div class="md-form">
                    <label for="windgustspeed" class="windgustspeed">Wind Gust
Speed</label>
```

```
<input type="text" class="form-control" id="windgustspeed"
name="windgustspeed">
</div>
</div>
<div class="col-md-6 my-2">
<div class="md-form">
<label for="winddir9am" class="winddir9am">Wind Direction
at 9am</label>
<select class="form-control" id="winddir9am"
name="winddir9am" aria-label="Wind Direction at 9am">
<option selected>Select Wind Direction at 9am</option>
<option value="1">N</option>
<option value="5">W</option>
<option value="10">S</option>
<option value="15">E</option>
<option value="2">NW</option>
<option value="9">NE</option>
<option value="7">SW</option>
<option value="13">SE</option>
<option value="0">NNW</option>
<option value="3">NNE</option>
<option value="8">SSW</option>
<option value="11">SSE</option>
<option value="4">WNW</option>
<option value="6">WSW</option>
<option value="12">ENE</option>
<option value="14">ESE</option>
</select>
```

```
          </div>
        </div>
        <div class="col-md-6 my-2">
          <div class="md-form">
            <label for="winddir3pm" class="winddir3pm">Wind Direction
at 3pm</label>
            <select class="form-control" id="winddir3pm"
name="winddir3pm" aria-label="Wind Direction at 3pm">
              <option selected>Select Wind Direction at 3pm</option>
              <option value="2">N</option>
              <option value="7">W</option>
              <option value="15">S</option>
              <option value="13">E</option>
              <option value="0">NW</option>
              <option value="10">NE</option>
              <option value="8">SW</option>
              <option value="12">SE</option>
              <option value="1">NNW</option>
              <option value="3">NNE</option>
              <option value="9">SSW</option>
              <option value="11">SSE</option>
              <option value="4">WNW</option>
              <option value="6">WSW</option>
              <option value="14">ENE</option>
              <option value="5">ESE</option>
            </select>
          </div>
```

```
            </div>
            <div class="col-md-6 my-2">
              <div class="md-form">
                <label for="windspeed9am" class="windspeed9am">Wind
Speed at 9am</label>
                <input type="text" class="form-control" id="windspeed9am"
name="windspeed9am">
              </div>
            </div>
            <div class="col-md-6 my-2">
              <div class="md-form">
                <label for="windspeed3pm" class="windspeed3pm">Wind
Speed at 3pm</label>
                <input type="text" class="form-control" id="windspeed3pm"
name="windspeed3pm">
              </div>
            </div>
            <div class="col-md-6 my-2">
              <div class="md-form">
                <label for="humidity9am" class="humidity9am">Humidity at
9am</label>
                <input type="text" class="form-control" id="humidity9am"
name="humidity9am">
              </div>
            </div>
            <div class="col-md-6 my-2">
              <div class="md-form">
                <label for="humidity3pm" class="humidity3pm">Humidity at
3pm</label>
```

```
                <input type="text" class="form-control" id="humidity3pm"
name="humidity3pm">
            </div>
        </div>
        <div class="col-md-6 my-2">
            <div class="md-form">
                <label for="pressure9am" class="pressure9am">Pressure at
9am</label>
                <input type="text" class="form-control" id="pressure9am"
name="pressure9am">
            </div>
        </div>
        <div class="col-md-6 my-2">
            <div class="md-form">
                <label for="pressure3pm" class="pressure3pm">Pressure at
3pm</label>
                <input type="text" class="form-control" id="pressure3pm"
name="pressure3pm">
            </div>
        </div>
        <div class="col-md-6 my-2">
            <div class="md-form">
                <label for="cloud9am" class="cloud9am">Cloud at
9am</label>
                <input type="text" class="form-control" id="cloud9am"
name="cloud9am">
            </div>
        </div>
        <div class="col-md-6 my-2">
```

```
                <div class="md-form">
                    <label for="cloud3pm" class="cloud3pm">Cloud at
3pm</label>
                    <input type="text" class="form-control" id="cloud3pm"
name="cloud3pm">
                </div>
            </div>
            <div class="col-md-6 my-2">
                <div class="md-form">
                    <label for="temp9am" class="temp9am">Temperature at
9am</label>
                    <input type="text" class="form-control" id="temp9am"
name="temp9am">
                </div>
            </div>
            <div class="col-md-6 my-2">
                <div class="md-form">
                    <label for="temp3pm" class="temp3pm">Temperature at
3pm</label>
                    <input type="text" class="form-control" id="temp3pm"
name="temp3pm">
                </div>
            </div>
            <div class="col-md-6 my-2">
                <div class="md-form">
                    <label for="raintoday" class="raintoday">Rain Today</label>
                    <select class="form-control" id="raintoday" name="raintoday"
aria-label="Rain Today">
                        <option selected>Select Rain Today</option>
```

```
                    <option value="0">No</option>
                    <option value="1">Yes</option>
                  </select>
                </div>
              </div>


              <div class="col-md-6 my-2">
                <div class="md-form">
                  <label for="date_month" class="date_month">Date
Month</label>
                  <input type="text" class="form-control" id="date_month"
name="date_month">
                </div>
              </div>
              <div class="col-md-6 my-2">
                <div class="md-form">
                  <label for="date_day" class="date_day">Date Day</label>
                  <input type="text" class="form-control" id="date_day"
name="date_day">
                </div>
              </div>
              <div class="col-12 mt-3">
                <button type="submit" class="btncolor">Predict
Rainfall</button>
              </div>
            </div>
          </form>
        </section>
```

</body>

</html>

❖ PREDICTOR.CSS

```
body {

    font-family: 'Poppins', sans-serif;
    height: 100vh;
    background-image: url("b11.jpg");
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment:fixed;
}
.box{
    display: flex;
    justify-content: center;
    align-items: center;
    margin: 20px;



}
.form {
    display:grid;
    border: 2px solid black;
    backdrop-filter: blur(2px);

    width: 50%;
    height: 10%;
    margin: 50px auto;
    padding: 20px 50px;
    box-shadow: 0 5px 11px 0 rgba(0,0,0,0.18),0 4px 15px 0
rgba(0,0,0,0.15);
    border-radius: 12px;
    display: grid;
```

```
    gap: 20px;
    color: white;

}
.form h1 {
    color: white;
}

button {


    font-size: 18px;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 10px;
    border-radius: 10px;
    margin-left: 40%;
    background-color:green;
    color: white;
}
.btncolor{
    background-color: green;
}
.date-l-t{
    display: flex;
    flex-direction: row;
}
```

❖ RAINY.HTML

```
  <!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;400;50
0;600;700;800;900&display=swap" rel="stylesheet">
   <link rel="stylesheet" href={{url_for('static',filename='style01.css')}}>
   <title>Rainy Day</title>
</head>
<body>
   <h1 style="text-align: center; font-size: 3 rem; font-weight:
bolder">TOMORROW WILL BE A RAINY DAY!</h1>
   <div class="rainyimg">
      <img src="{{ url_for('static', filename='images/rainyday.jpg') }}"
style=" max-height:20%; max-width:75%;padding-left:12%;padding-
right:15%">
   </div>
  </div>
</body>
</html>
```

- **STYLE.CSS**

```
body {
   background-color: white;
   font-family: 'poppins',sans-serif;
}

h2{
   font-size: 2 rem;
   font-weight: bold;
}
```

- **SUNNY.HTML**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;400;50
0;600;700;800;900&display=swap" rel="stylesheet">
   <link rel="stylesheet" href={{url_for('static',filename='style01.css')}}>
   <title>Rainy Day</title>
</head>
<body>
   <h1 style="text-align: center; font-size: 3 rem; font-weight:
bolder">TOMORROW WILL BE A SUNNY DAY!</h1>
   <div class="rainyimg">
      <img src="{{ url_for('static', filename='images/sunnn_day.jpg') }}"
style=" height:10%; width:60%;padding-left:20%;padding-right:15%">
```

```
</div>
</div>
</body>
</html>
```

- **STYLE.CSS**

```
body {
   background-color: white;
   font-family: 'poppins',sans-serif;
}

h2{
   font-size: 2 rem;
   font-weight: bold;
}
```

- **APP.PY**

```
import numpy as np
import pickle
import joblib
import matplotlib.pyplot as plt
import time
import pandas as pd
import os
from flask import Flask, request, jsonify, render_template
import sklearn
```

```
app =
Flask(_name_,static_folder='static',template_folder='templates
')
model = pickle.load(open('rainfall.pkl1', 'rb'))
scale = pickle.load(open('scale.pkl1', 'rb'))

@app.route("/")
def home():
   return render_template('home (1).html')

@app.route('/predict', methods=["POST", "GET"])
def predict():
   # Fetch form data
   input_feature = [request.form.get(key) for key in
request.form.keys() if key != 'date']

   # Print for debugging
   print(f"Input features: {input_feature}")



   features_values = [np.array(input_feature)]

   names = ['Location', 'MinTemp', 'MaxTemp',
'Rainfall','Evaporation','Sunshine', 'WindGustDir',
          'WindGustSpeed','WindDir9am', 'WindDir3pm',
'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
'Humidity3pm',
          'Pressure9am', 'Pressure3pm', 'Cloud9am','Cloud3pm',
'Temp9am', 'Temp3pm', 'RainToday', 'Date_month',
'Date_day']

   data = pd.DataFrame(features_values, columns=names)
   print(data.columns)
```

```
    # Print for debugging
    print(f"Data before scaling: \n{data}")
    data = scale.transform(data)  # Use transform instead of
fit_transform
    data = pd.DataFrame(data, columns=names)

    # Print for debugging
    print(f"Data after scaling: \n{data}")

    # Predictions using the loaded model files
    prediction = model.predict(data)
    pred_prob = model.predict_proba(data)

    print(f"Prediction: {prediction}")
    if prediction[0] == "YES":
        return render_template("sunny.html")
    else:
        return render_template("rainy.html")

if _name_ == "_main_":
    app.run(debug=True)
```

CODE SNIPPETS

MODEL BUILDING

Rainfall Prediction

```
# @title Rainfall Prediction
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import preprocessing
import scipy.stats as stat
```

```
[ ] df = pd.read_csv("/content/weatherAUS.csv")
    pd.set_option("display.max_columns", None)
    df.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	WNW	20.0	24.0	71.0	22.0	1007.7	1007.1	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	WSW	4.0	22.0	44.0	25.0	1010.6	1007.8	N
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	WSW	19.0	26.0	38.0	30.0	1007.6	1008.7	N
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	E	11.0	9.0	45.0	16.0	1017.6	1012.8	N
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	NW	7.0	20.0	82.0	33.0	1010.8	1006.0	

```
[ ] df.shape

(145460, 23)
```

```
[ ] numerical_feature = [feature for feature in df.columns if df[feature].dtypes != 'O']
    discrete_feature = [feature for feature in numerical_feature if len(df[feature].unique()) < 25]
    continuous_feature = [feature for feature in numerical_feature if feature not in discrete_feature]
    categorical_feature = [feature for feature in df.columns if feature not in numerical_feature]

    print("Numerical Features Count {}".format(len(numerical_feature)))
    print("Discrete Features Count {}".format(len(discrete_feature)))
    print("Continuous Features Count {}".format(len(continuous_feature)))
    print("Categorical Features Count {}".format(len(categorical_feature)))

Numerical Features Count 16
Discrete Features Count 2
Continuous Features Count 14
Categorical Features Count 7
```

```
[ ] print(numerical_feature)

['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
```

```
[ ] print(discrete_feature)

['Cloud9am', 'Cloud3pm']
```

Handling Missing Values

```
# @title Handling Missing Values
df.isnull().sum()

Date                0
Location            0
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Evaporation     62790
Sunshine        69835
WindGustDir     10326
WindGustSpeed   10263
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am      2654
Humidity3pm      4507
Pressure9am     15065
Pressure3pm     15028
Cloud9am        55888
Cloud3pm        59358
Temp9am          1767
Temp3pm          3609
RainToday        3261
RainTomorrow     3267
dtype: int64
```

```
[ ] df.dropna(inplace=True)
```

Using Random sample imputation for handling missing values of features 'Evaporation' and 'Sunshine'

```
[ ] # @title Using Random sample imputation for handling missing values of features 'Evaporation' and 'Sunshine'
    def randomsampleimputation(df,feature):
        df[feature] = df[feature]
        random_sample = df[feature].dropna().sample(df[feature].isnull().sum(), random_state = 0)
        random_sample.index = df[df[feature].isnull()].index
        df.loc[df[feature].isnull(), feature] = random_sample
```

```
randomsampleimputation(df, "Evaporation")
randomsampleimputation(df, "Sunshine")
```

```
df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9a
6049	2009-01-01	Cobar	17.9	35.2	0.0	12.0	12.3	SSW	48.0	ENE	SW	6.0	20.0	20.0	13.0	1006.3	1004.4	2.0	5.0	26.
6050	2009-01-02	Cobar	18.4	28.9	0.0	14.8	13.0	S	37.0	SSE	SSE	19.0	19.0	30.0	8.0	1012.9	1012.1	1.0	1.0	20.
6052	2009-01-04	Cobar	19.4	37.6	0.0	10.8	10.6	NNE	46.0	NNE	NNW	30.0	15.0	42.0	22.0	1012.3	1009.2	1.0	6.0	28.
6053	2009-01-05	Cobar	21.9	38.4	0.0	11.4	12.2	WNW	31.0	WNW	WSW	6.0	6.0	37.0	22.0	1012.7	1009.1	1.0	5.0	29.
6054	2009-01-06	Cobar	24.2	41.0	0.0	11.2	8.4	WNW	35.0	NW	WNW	17.0	13.0	19.0	15.0	1010.7	1007.4	1.0	6.0	33.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
142298	2017-06-20	Darwin	19.3	33.4	0.0	6.0	11.0	ENE	35.0	SE	NE	9.0	20.0	63.0	32.0	1013.9	1010.5	0.0	1.0	24.
142299	2017-06-21	Darwin	21.2	32.6	0.0	7.6	8.6	E	37.0	SE	SE	13.0	11.0	56.0	28.0	1014.6	1011.2	7.0	0.0	24.
142300	2017-06-22	Darwin	20.7	32.8	0.0	5.6	11.0	E	33.0	E	W	17.0	11.0	46.0	23.0	1015.3	1011.8	0.0	0.0	24.
142301	2017-06-23	Darwin	19.5	31.8	0.0	6.2	10.6	ESE	26.0	SE	NNW	9.0	17.0	62.0	58.0	1014.9	1010.7	1.0	1.0	24.
142302	2017-06-24	Darwin	20.2	31.7	0.0	5.6	10.7	ENE	30.0	ENE	NNW	15.0	7.0	73.0	32.0	1013.9	1009.7	6.0	5.0	25.

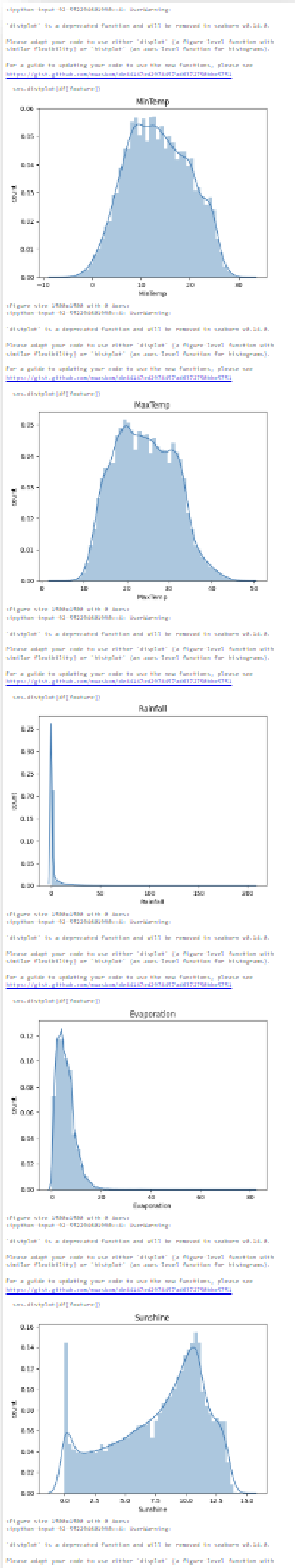
56420 rows × 23 columns

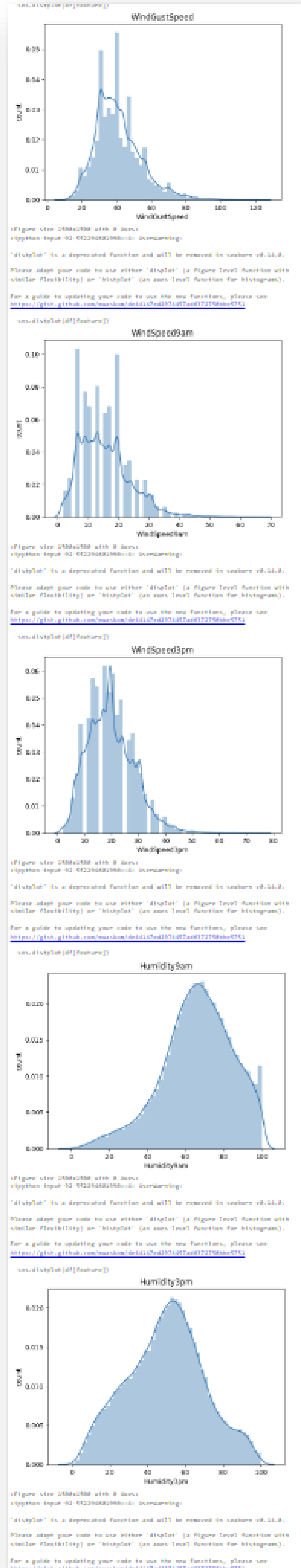
```
df.isnull().sum()
```

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

Analyzing each continuous feature

```
# @title Analyzing each continuous feature
for feature in continuous_feature:
    data = df.copy()
    sns.distplot(df[feature])
    plt.xlabel(feature)
    plt.ylabel('count')
    plt.title(feature)
    plt.figure(figsize = (15,15))
    plt.show()
```


MinTemp
MaxTemp
Rainfall
Evaporation
Sunshine


WindGustSpeed
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm

Replacing the null values of the remaining continuous features by median

```
# @title Replacing the null values of the remaining continuous features by median
for feature in continuous_feature:
    if(df[feature].isnull().sum()*100/len(df))>0:
        df[feature] = df[feature].fillna(df[feature].median())
```

```
df.isnull().sum()
```

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

Replacing the null values of the discrete features by mode

```
# @title Replacing the null values of the discrete features by mode
def mode_nan(df,variable):
    mode=df[variable].value_counts().index[0]
    df[variable].fillna(mode,inplace=True)
mode_nan(df,"Cloud9am")
mode_nan(df,"Cloud3pm")
```

```
df.isnull().sum()
```

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

Handling categorical features using One Hot Encoding

```
# @title Handling categorical features using One Hot Encoding
df["RainToday"] = pd.get_dummies(df["RainToday"], drop_first = True)
df["RainTomorrow"] = pd.get_dummies(df["RainTomorrow"], drop_first = True)
df
```

Show hidden output

```
# prompt: Using dataframe df: convert RainToday and RainTomorrowcolumns true=1,false=0

df['RainToday'] = df['RainToday'].apply(lambda x: 1 if x == True else 0)
df['RainTomorrow'] = df['RainTomorrow'].apply(lambda x: 1 if x == True else 0)
```

Grouping categorical features by RainTomorrow

```
# @title Grouping categorical features by 'RainTomorrow'
for feature in categorical_feature:
    print(feature, (df.groupby([feature])["RainTomorrow"].mean().sort_values(ascending = False)).index)
```

```
Date Index(['2007-11-01', '2008-02-03', '2008-01-30', '2008-01-19', '2008-01-18',
       '2008-11-19', '2008-01-16', '2017-03-21', '2008-01-12', '2008-03-24',
       ...
       '2008-06-13', '2008-06-14', '2008-06-15', '2008-06-16', '2008-06-17',
       '2008-06-18', '2008-06-20', '2008-06-21', '2008-06-22', '2008-12-24'],
      dtype='object', name='Date', length=3416)
Location Index(['Portland', 'CoffsHarbour', 'Cairns', 'NorfolkIsland', 'MountGambier',
       'Williamtown', 'Darwin', 'Watsonia', 'SydneyAirport', 'Melbourne',
       'Sydney', 'Hobart', 'Brisbane', 'MelbourneAirport', 'Sale', 'Perth',
       'Canberra', 'Nuriootpa', 'PerthAirport', 'WaggaWagga', 'Townsville',
       'Moree', 'Cobar', 'Mildura', 'AliceSprings', 'Woomera'],
      dtype='object', name='Location')
WindGustDir Index(['WNW', 'NW', 'W', 'NNW', 'N', 'WSW', 'NNE', 'S', 'SSW', 'SSE', 'SW',
       'SE', 'NE', 'ESE', 'E', 'ENE'],
      dtype='object', name='WindGustDir')
WindDir9am Index(['N', 'NNW', 'NNE', 'W', 'NW', 'WNW', 'WSW', 'SW', 'NE', 'S', 'SSE',
       'SSW', 'ENE', 'SE', 'ESE', 'E'],
      dtype='object', name='WindDir9am')
WindDir3pm Index(['WNW', 'NW', 'N', 'NNW', 'W', 'NNE', 'WSW', 'SSE', 'SSW', 'S', 'SE',
       'NE', 'ESE', 'E', 'SW', 'ENE'],
      dtype='object', name='WindDir3pm')
RainToday Index([1, 0], dtype='int64', name='RainToday')
RainTomorrow Index([1, 0], dtype='int64', name='RainTomorrow')
```

Replacing the null values of the categorical features "WindGustDir", "WindDir9am" and "WindDir3pm"using Label Encoding,to avoid curse of dimensionality

```
# @title Replacing the null values of the categorical features "WindGustDir", "WindDir9am" and "WindDir3pm"using Label Encoding,to avoid curse of dimensionality
windgustdir = {'NNW':0, 'NW':1, 'WNW':2, 'N':3, 'W':4, 'WSW':5, 'NNE':6, 'S':7, 'SSW':8, 'SW':9, 'SSE':10,
       'NE':11, 'SE':12, 'ESE':13, 'ENE':14, 'E':15}
winddir9am = {'NNW':0, 'N':1, 'NW':2, 'NNE':3, 'WNW':4, 'W':5, 'WSW':6, 'SW':7, 'SSW':8, 'NE':9, 'S':10,
       'SSE':11, 'ENE':12, 'SE':13, 'ESE':14, 'E':15}
winddir3pm = {'NW':0, 'NNW':1, 'N':2, 'WNW':3, 'W':4, 'NNE':5, 'WSW':6, 'SSW':7, 'S':8, 'SW':9, 'SE':10,
       'NE':11, 'SSE':12, 'ENE':13, 'E':14, 'ESE':15}
df["WindGustDir"] = df["WindGustDir"].map(windgustdir)
df["WindDir9am"] = df["WindDir9am"].map(winddir9am)
df["WindDir3pm"] = df["WindDir3pm"].map(winddir3pm)
```

```
df["WindGustDir"] = df["WindGustDir"].fillna(df["WindGustDir"].value_counts().index[0])
df["WindDir9am"] = df["WindDir9am"].fillna(df["WindDir9am"].value_counts().index[0])
df["WindDir3pm"] = df["WindDir3pm"].fillna(df["WindDir3pm"].value_counts().index[0])
```

+ Code + Text

```
df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainTo
6049	2009-01-01	Cobar	17.9	35.2	0.0	12.0	12.3	8	48.0	12	9	6.0	20.0	20.0	13.0	1006.3	1004.4	2.0	5.0	26.6	33.4	
6050	2009-01-02	Cobar	18.4	28.9	0.0	14.8	13.0	7	37.0	11	12	19.0	19.0	30.0	8.0	1012.9	1012.1	1.0	1.0	20.3	27.0	
6052	2009-01-04	Cobar	19.4	37.6	0.0	10.8	10.6	6	46.0	3	1	30.0	15.0	42.0	22.0	1012.3	1009.2	1.0	6.0	28.7	34.9	
6053	2009-01-05	Cobar	21.9	38.4	0.0	11.4	12.2	2	31.0	4	6	6.0	6.0	37.0	22.0	1012.7	1009.1	1.0	5.0	29.1	35.6	
6054	2009-01-06	Cobar	24.2	41.0	0.0	11.2	8.4	2	35.0	2	3	17.0	13.0	19.0	15.0	1010.7	1007.4	1.0	6.0	33.6	37.6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
142298	2017-06-20	Darwin	19.3	33.4	0.0	6.0	11.0	14	35.0	13	11	9.0	20.0	63.0	32.0	1013.9	1010.5	0.0	1.0	24.5	32.3	
142299	2017-06-21	Darwin	21.2	32.6	0.0	7.6	8.6	15	37.0	13	10	13.0	11.0	56.0	28.0	1014.6	1011.2	7.0	0.0	24.8	32.0	
142300	2017-06-22	Darwin	20.7	32.8	0.0	5.6	11.0	15	33.0	15	4	17.0	11.0	46.0	23.0	1015.3	1011.8	0.0	0.0	24.8	32.1	
142301	2017-06-23	Darwin	19.5	31.8	0.0	6.2	10.6	13	26.0	13	1	9.0	17.0	62.0	58.0	1014.9	1010.7	1.0	1.0	24.8	29.2	
142302	2017-06-24	Darwin	20.2	31.7	0.0	5.6	10.7	14	30.0	12	1	15.0	7.0	73.0	32.0	1013.9	1009.7	6.0	5.0	25.4	31.0	

56420 rows × 23 columns

```
df.isnull().sum()
```

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

Performing Label Encoding on "Location"

```
# @title Performing Label Encoding on "Location"
df1 = df.groupby(["Location"])["RainTomorrow"].value_counts().sort_values().unstack()
```

```
df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainTo
6049	2009-01-01	Cobar	17.9	35.2	0.0	12.0	12.3	8	48.0	12	9	6.0	20.0	20.0	13.0	1006.3	1004.4	2.0	5.0	26.6	33.4	
6050	2009-01-02	Cobar	18.4	28.9	0.0	14.8	13.0	7	37.0	11	12	19.0	19.0	30.0	8.0	1012.9	1012.1	1.0	1.0	20.3	27.0	
6052	2009-01-04	Cobar	19.4	37.6	0.0	10.8	10.6	6	46.0	3	1	30.0	15.0	42.0	22.0	1012.3	1009.2	1.0	6.0	28.7	34.9	

```
df1[1].sort_values(ascending = False)
```

```
Location
Darwin              789
Cairns              751
Portland            746
NorfolkIsland       743
MountGambier        732
SydneyAirport       715
Watsonia            684
Brisbane            642
MelbourneAirport    636
Perth               616
PerthAirport        553
Melbourne           471
Hobart              468
CoffsHarbour        431
WaggaWagga          427
Sydney              416
Townsville          394
Nuriootpa           389
Sale                359
Williamtown         320
Mildura             289
Moree               258
Canberra            219
AliceSprings        187
Woomera             129
Cobar                63
Name: 1, dtype: int64
```

```
df1[1].sort_values(ascending = False).index
```

```
Index(['Darwin', 'Cairns', 'Portland', 'NorfolkIsland', 'MountGambier',
       'SydneyAirport', 'Watsonia', 'Brisbane', 'MelbourneAirport', 'Perth',
       'PerthAirport', 'Melbourne', 'Hobart', 'CoffsHarbour', 'WaggaWagga',
       'Sydney', 'Townsville', 'Nuriootpa', 'Sale', 'Williamtown', 'Mildura',
       'Moree', 'Canberra', 'AliceSprings', 'Woomera', 'Cobar'],
      dtype='object', name='Location')
```

```
len(df1[1].sort_values(ascending = False).index)
```

```
26
```

```
[ ] location = {'Portland':1, 'Cairns':2, 'Walpole':3, 'Dartmoor':4, 'MountGambier':5,
            'NorfolkIsland':6, 'Albany':7, 'Witchcliffe':8, 'CoffsHarbour':9, 'Sydney':10,
            'Darwin':11, 'MountGinini':12, 'NorahHead':13, 'Ballarat':14, 'GoldCoast':15,
            'SydneyAirport':16, 'Hobart':17, 'Watsonia':18, 'Newcastle':19, 'Wollongong':20,
            'Brisbane':21, 'Williamtown':22, 'Launceston':23, 'Adelaide':24, 'MelbourneAirport':25,
            'Perth':26, 'Sale':27, 'Melbourne':28, 'Canberra':29, 'Albury':30, 'Penrith':31,
            'Nuriootpa':32, 'BadgerysCreek':33, 'Tuggeranong':34, 'PerthAirport':35, 'Bendigo':36,
            'Richmond':37, 'WaggaWagga':38, 'Townsville':39, 'PearceRAAF':40, 'SalmonGums':41,
            'Moree':42, 'Cobar':43, 'Mildura':44, 'Katherine':45, 'AliceSprings':46, 'Nhil':47,
            'Woomera':48, 'Uluru':49}
    df["Location"] = df["Location"].map(location)
```

```
[ ] df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainT
6049	2009-01-01	43	17.9	35.2	0.0	12.0	12.3	8	48.0	12	9	6.0	20.0	20.0	13.0	1006.3	1004.4	2.0	5.0	26.6	33.4	
6050	2009-01-02	43	18.4	28.9	0.0	14.8	13.0	7	37.0	11	12	19.0	19.0	30.0	8.0	1012.9	1012.1	1.0	1.0	20.3	27.0	
6052	2009-01-04	43	19.4	37.6	0.0	10.8	10.6	6	46.0	3	1	30.0	15.0	42.0	22.0	1012.3	1009.2	1.0	6.0	28.7	34.9	
6053	2009-01-05	43	21.9	38.4	0.0	11.4	12.2	2	31.0	4	6	6.0	6.0	37.0	22.0	1012.7	1009.1	1.0	5.0	29.1	35.6	
6054	2009-01-06	43	24.2	41.0	0.0	11.2	8.4	2	35.0	2	3	17.0	13.0	19.0	15.0	1010.7	1007.4	1.0	6.0	33.6	37.6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
142298	2017-06-20	11	19.3	33.4	0.0	6.0	11.0	14	35.0	13	11	9.0	20.0	63.0	32.0	1013.9	1010.5	0.0	1.0	24.5	32.3	
142299	2017-06-21	11	21.2	32.6	0.0	7.6	8.6	15	37.0	13	10	13.0	11.0	56.0	28.0	1014.6	1011.2	7.0	0.0	24.8	32.0	
142300	2017-06-22	11	20.7	32.8	0.0	5.6	11.0	15	33.0	15	4	17.0	11.0	46.0	23.0	1015.3	1011.8	0.0	0.0	24.8	32.1	
142301	2017-06-23	11	19.5	31.8	0.0	6.2	10.6	13	26.0	13	1	9.0	17.0	62.0	58.0	1014.9	1010.7	1.0	1.0	24.8	29.2	
[illegible]	2017-																					

```
import pandas as pd # Import pandas for datetime operations

# Convert 'Date' column to datetime objects
df['Date'] = pd.to_datetime(df['Date'])

# Now you can extract month, day, and year
df['Date_month'] = df['Date'].dt.month
df['Date_day'] = df['Date'].dt.day
```

```
[ ] df
```

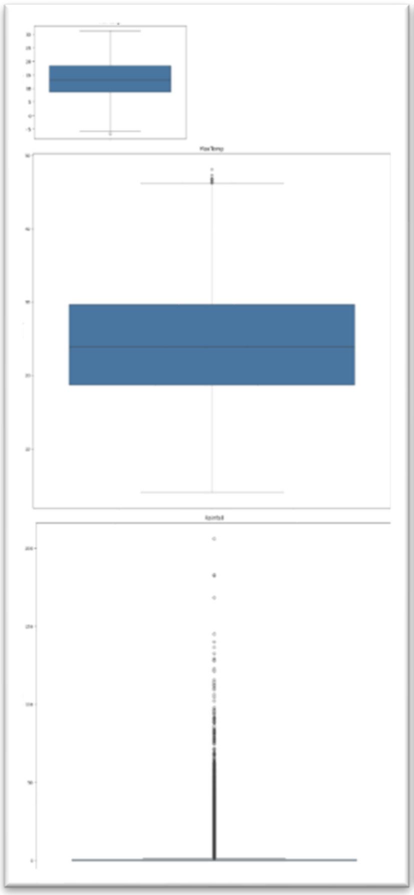
	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainTo
6049	2009-01-01	43	17.9	35.2	0.0	12.0	12.3	8	48.0	12	9	6.0	20.0	20.0	13.0	1006.3	1004.4	2.0	5.0	26.6	33.4	
6050	2009-01-02	43	18.4	28.9	0.0	14.8	13.0	7	37.0	11	12	19.0	19.0	30.0	8.0	1012.9	1012.1	1.0	1.0	20.3	27.0	
6052	2009-01-04	43	19.4	37.6	0.0	10.8	10.6	6	46.0	3	1	30.0	15.0	42.0	22.0	1012.3	1009.2	1.0	6.0	28.7	34.9	
6053	2009-01-05	43	21.9	38.4	0.0	11.4	12.2	2	31.0	4	6	6.0	6.0	37.0	22.0	1012.7	1009.1	1.0	5.0	29.1	35.6	
6054	2009-01-06	43	24.2	41.0	0.0	11.2	8.4	2	35.0	2	3	17.0	13.0	19.0	15.0	1010.7	1007.4	1.0	6.0	33.6	37.6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
142298	2017-06-20	11	19.3	33.4	0.0	6.0	11.0	14	35.0	13	11	9.0	20.0	63.0	32.0	1013.9	1010.5	0.0	1.0	24.5	32.3	
142299	2017-06-21	11	21.2	32.6	0.0	7.6	8.6	15	37.0	13	10	13.0	11.0	56.0	28.0	1014.6	1011.2	7.0	0.0	24.8	32.0	
142300	2017-06-22	11	20.7	32.8	0.0	5.6	11.0	15	33.0	15	4	17.0	11.0	46.0	23.0	1015.3	1011.8	0.0	0.0	24.8	32.1	
142301	2017-06-23	11	19.5	31.8	0.0	6.2	10.6	13	26.0	13	1	9.0	17.0	62.0	58.0	1014.9	1010.7	1.0	1.0	24.8	29.2	
142302	2017-06-24	11	20.2	31.7	0.0	5.6	10.7	14	30.0	12	1	15.0	7.0	73.0	32.0	1013.9	1009.7	6.0	5.0	25.4	31.0	

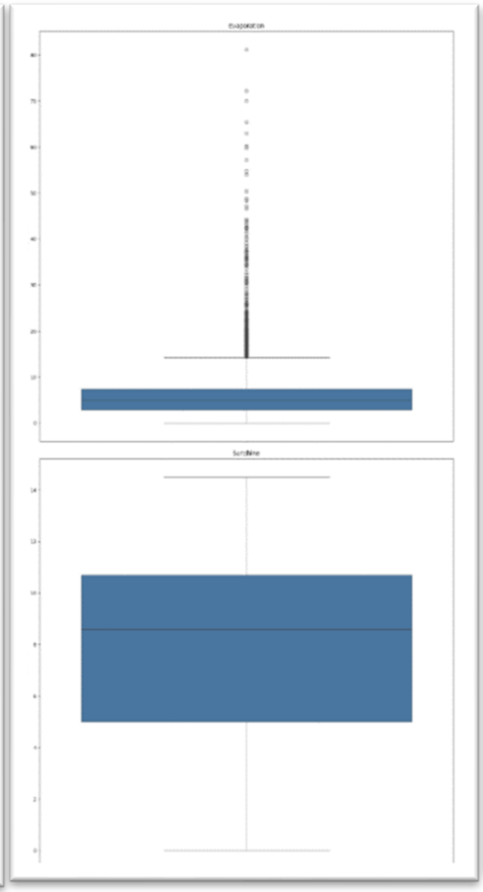
56420 rows × 25 columns

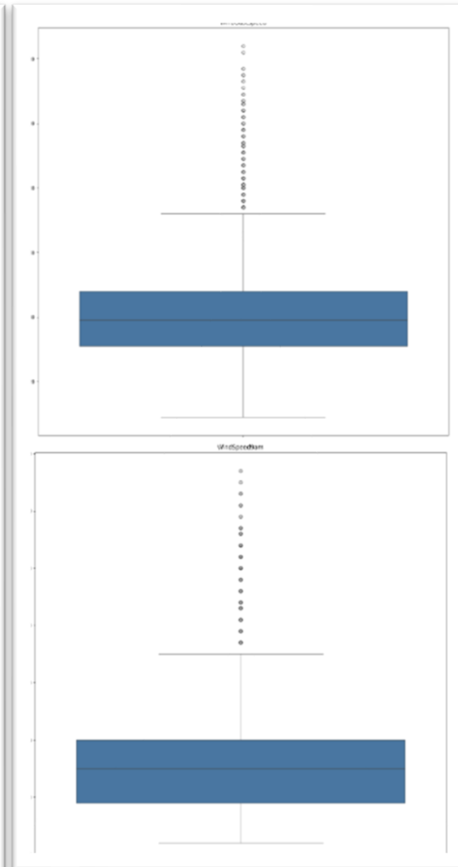
	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am
Date	1.000000	-0.055381	0.048743	0.030813	-0.014134	0.043027	0.008583	0.004385	-0.013787	-0.012689	0.013188	-0.014623
Location	-0.055381	1.000000	-0.146144	0.139633	-0.105028	0.250813	0.165778	0.056351	-0.032248	0.106122	-0.026189	-0.018413
MinTemp	0.048743	-0.146144	1.000000	0.749687	0.110800	0.506764	0.076087	0.222044	0.118054	0.195342	0.170180	0.108522
MaxTemp	0.030813	0.139633	0.749687	1.000000	-0.069981	0.650964	0.463913	0.204657	0.035531	0.230890	0.092713	-0.008171
Rainfall	-0.014134	-0.105028	0.110800	-0.069981	1.000000	-0.077239	-0.246379	-0.015310	0.106308	-0.008678	-0.005006	0.050584
Evaporation	0.043027	0.250813	0.506764	0.650964	-0.077239	1.000000	0.369250	0.168033	0.209566	0.190507	0.110799	0.193154
Sunshine	0.008583	0.165778	0.076087	0.463913	-0.246379	0.369250	1.000000	0.138268	-0.052422	0.164061	0.077414	-0.013842
WindGustDir	0.004385	0.056351	0.222044	0.204657	-0.015310	0.168033	0.138268	1.000000	-0.187983	0.551234	0.640969	-0.056210
WindGustSpeed	-0.013787	-0.032248	0.118054	0.035531	0.106308	0.209566	-0.052422	-0.187983	1.000000	-0.190865	-0.185578	0.608852
WindDir9am	-0.012689	0.106122	0.195342	0.230890	-0.008678	0.190507	0.164061	0.551234	-0.190865	1.000000	0.431097	-0.050612
WindDir3pm	0.013188	-0.026189	0.170180	0.092713	-0.005006	0.110799	0.077414	0.640969	-0.185578	0.431097	1.000000	-0.052322
WindSpeed9am	-0.014623	-0.018413	0.108522	-0.008171	0.050584	0.193154	-0.013842	-0.056210	0.608852	-0.050612	-0.052322	1.000000
WindSpeed3pm	-0.023544	-0.114013	0.137273	0.012266	0.044112	0.124345	0.029200	-0.102208	0.685236	-0.140853	-0.080973	0.502226
Humidity9am	-0.023821	-0.285646	-0.174991	-0.495781	0.263625	-0.554232	-0.500343	-0.074697	-0.193410	-0.097566	-0.027596	-0.236795
Humidity3pm	0.002700	-0.440490	0.070827	-0.448116	0.277625	-0.422436	-0.629299	-0.021968	-0.042653	-0.115076	0.064196	-0.058449
Pressure9am	0.036972	0.069021	-0.475661	-0.350980	-0.180606	-0.297319	0.043330	0.176116	-0.430363	0.132545	0.189341	-0.201518
Pressure3pm	0.036870	0.027490	-0.495577	-0.451691	-0.137302	-0.325954	-0.020632	0.153787	-0.383683	0.105269	0.196007	-0.155484
Cloud9am	-0.003380	-0.213413	0.106177	-0.263834	0.217169	-0.199809	-0.677939	-0.075962	0.088129	-0.090312	-0.028697	0.034908
Cloud3pm	-0.015967	-0.165286	0.042173	-0.257752	0.191433	-0.202366	-0.702022	-0.133115	0.131590	-0.148962	-0.098134	0.062507
Temp9am	0.043517	-0.039302	0.906295	0.893387	0.013713	0.593122	0.289038	0.235351	0.085520	0.211529	0.155901	0.053749
Temp3pm	0.026623	0.152084	0.727222	0.984841	-0.074627	0.630727	0.486129	0.214602	-0.000382	0.247186	0.091679	-0.018357
RainToday	-0.004840	-0.162697	0.048953	-0.221810	0.550515	-0.218079	-0.328904	-0.070255	0.146276	-0.075836	-0.036234	0.083125
RainTomorrow	-0.005977	-0.161183	0.087428	-0.147467	0.254342	-0.130002	-0.453407	-0.110127	0.233158	-0.129472	-0.098542	0.083904
Date_month	0.009428	-0.007087	-0.191405	-0.180873	-0.033498	-0.046118	0.030639	-0.087366	0.043544	-0.087391	-0.084072	0.048618
Date_day	0.008595	-0.003311	0.000160	0.000365	0.008651	-0.012605	-0.002696	-0.020446	-0.012689	-0.018309	-0.022893	-0.008874

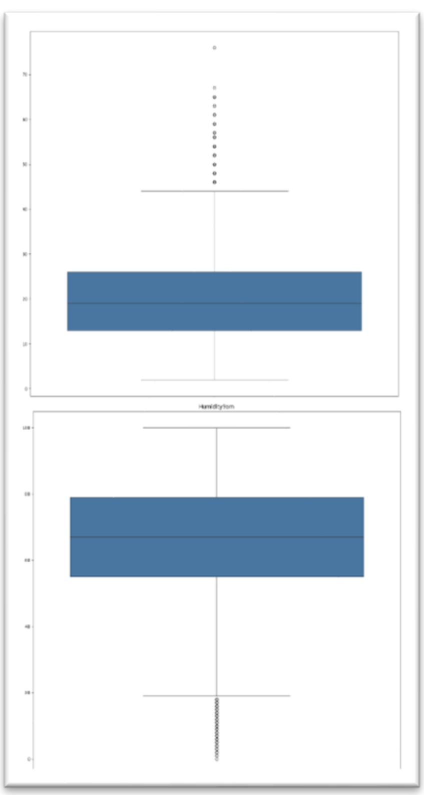
	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow	Date_month	Date_day
Date	-0.023544	-0.023821	0.002700	0.036972	0.036870	-0.003380	-0.015967	0.043517	0.026623	-0.004840	-0.005977	0.009428	0.008595
Location	-0.114013	-0.285646	-0.440490	0.069021	0.027490	-0.213413	-0.165286	-0.039302	0.152084	-0.162697	-0.161183	-0.007087	-0.003311
MinTemp	0.137273	-0.174991	0.070827	-0.475661	-0.495577	0.106177	0.042173	0.906295	0.727222	0.048953	0.087428	-0.191405	0.000160
MaxTemp	0.012266	-0.495781	-0.448116	-0.350980	-0.451691	-0.263834	-0.257752	0.893387	0.984841	-0.221810	-0.147467	-0.180873	0.000365
Rainfall	0.044112	0.263625	0.277625	-0.180606	-0.137302	0.217169	0.191433	0.013713	-0.074627	0.550515	0.254342	-0.033498	0.008651
Evaporation	0.124345	-0.554232	-0.422436	-0.297319	-0.325954	-0.199809	-0.202366	0.593122	0.630727	-0.218079	-0.130002	-0.046118	-0.012605
Sunshine	0.029200	-0.500343	-0.629299	0.043330	-0.020632	-0.677939	-0.702022	0.289038	0.486129	-0.328904	-0.453407	0.030639	-0.002696
WindGustDir	-0.102208	-0.074697	-0.021968	0.176116	0.153787	-0.075962	-0.133115	0.235351	0.214602	-0.070255	-0.110127	-0.087366	-0.020446
WindGustSpeed	0.685236	-0.193410	-0.042653	-0.430363	-0.383683	0.088129	0.131590	0.085520	-0.000382	0.146276	0.233158	0.043544	-0.012689
WindDir9am	-0.140853	-0.097566	-0.115076	0.132545	0.105269	-0.090312	-0.148962	0.211529	0.247186	-0.075836	-0.129472	-0.087391	-0.018309
WindDir3pm	-0.080973	-0.027596	0.064196	0.189341	0.196007	-0.028697	-0.098134	0.155901	0.091679	-0.036234	-0.098542	-0.084072	-0.022893
WindSpeed9am	0.502226	-0.236795	-0.058449	-0.201518	-0.155484	0.034908	0.062507	0.053749	-0.018357	0.083125	0.083904	0.048618	-0.008874
WindSpeed3pm	1.000000	-0.100626	0.031843	-0.293155	-0.252095	0.068224	0.041475	0.114043	-0.009436	0.085627	0.088862	0.055328	-0.011822
Humidity9am	-0.100626	1.000000	0.685697	0.114575	0.172972	0.438962	0.348707	-0.423598	-0.487758	0.379451	0.271033	-0.092247	0.013172
Humidity3pm	0.031843	0.685697	1.000000	-0.063454	0.024109	0.509223	0.510996	-0.151614	-0.497245	0.385440	0.455358	-0.026354	0.011936
Pressure9am	-0.293155	0.114575	-0.063454	1.000000	0.961538	-0.150427	-0.166334	-0.443410	-0.310774	-0.186848	-0.254818	0.058443	-0.014815
Pressure3pm	-0.252095	0.172972	0.024109	0.961538	1.000000	-0.081791	-0.103173	-0.500509	-0.421318	-0.104103	-0.230418	0.046967	-0.013833
Cloud9am	0.068224	0.438962	0.509223	-0.150427	-0.081791	1.000000	0.614380	-0.109591	-0.281213	0.297794	0.323972	-0.006727	0.008647
Cloud3pm	0.041475	0.348707	0.510996	-0.166334	-0.103173	0.614380	1.000000	-0.107885	-0.297230	0.272034	0.388574	-0.010735	-0.000687
Temp9am	0.114043	-0.423598	-0.151614	-0.443410	-0.500509	-0.109591	-0.107885	1.000000	0.870620	-0.096946	-0.018179	-0.125118	-0.000566
Temp3pm	-0.009436	-0.487758	-0.497245	-0.310774	-0.421318	-0.281213	-0.297230	0.870620	1.000000	-0.228515	-0.183586	-0.186431	-0.000999
RainToday	0.085627	0.379451	0.385440	-0.186848	-0.104103	0.297794	0.272034	-0.096946	-0.228515	1.000000	0.309098	0.002787	-0.000863
RainTomorrow	0.088862	0.271033	0.455358	-0.254818	-0.230418	0.323972	0.388574	-0.018179	-0.183586	0.309098	1.000000	0.001046	0.001389
Date_month	0.055328	-0.092247	-0.026354	0.058443	0.046967	-0.006727	-0.010735	-0.125118	-0.186431	0.002787	0.001046	1.000000	0.008643
Date_day	-0.011822	0.013172	0.011936	-0.014815	-0.013833	0.008647	-0.000687	-0.000566	-0.000999	-0.000863	0.001389	0.008643	1.000000

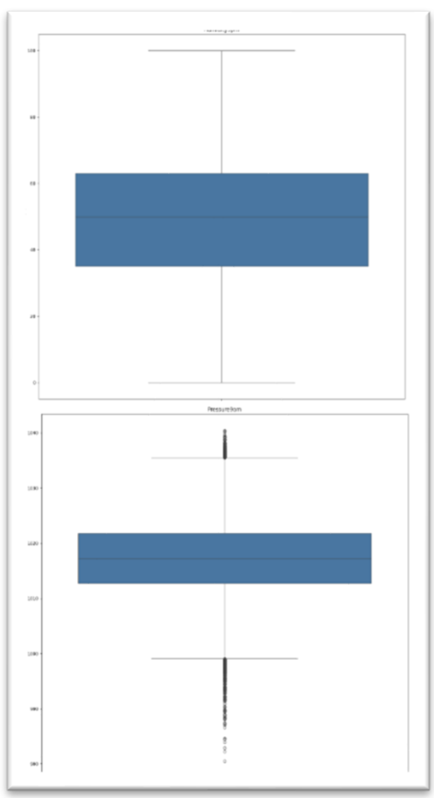
```
for feature in continuous_feature:
    data=df.copy()
    sns.boxplot(data[feature])
    plt.title(feature)
    plt.figure(figsize=(15,15))
```

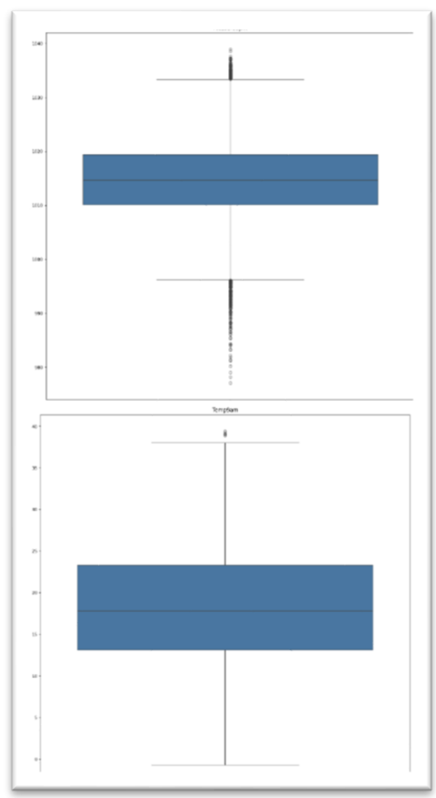






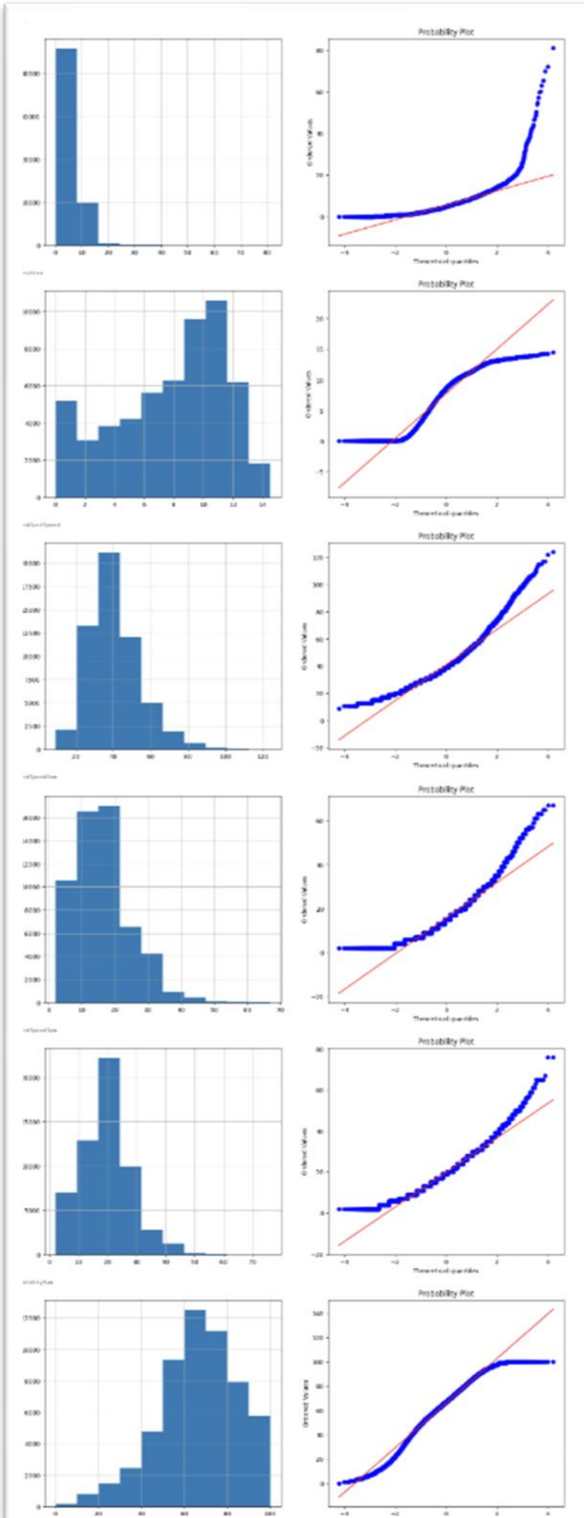


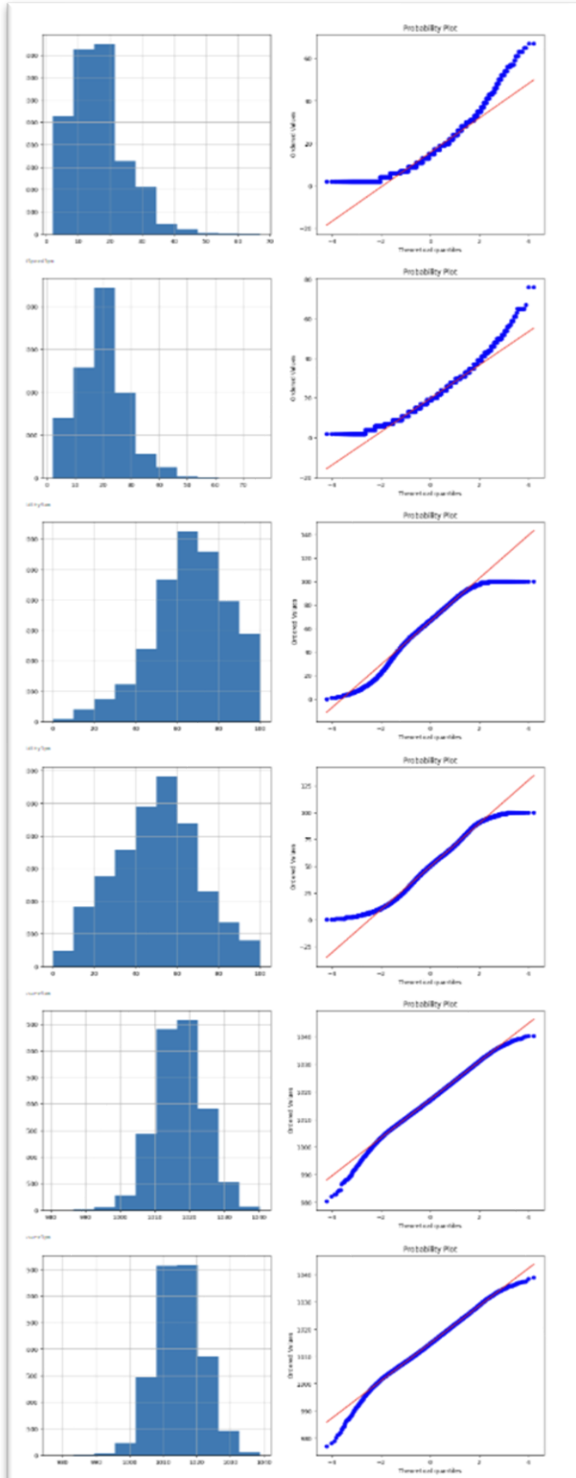




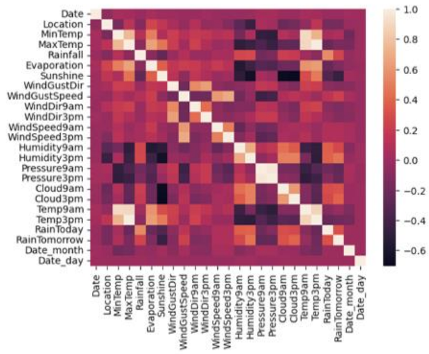
```python
# @title Feature Transformation
def qq_plots(df, variable):
    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df[variable].hist()
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

```python
for feature in continuous_feature:
    print(feature)
    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df[feature].hist()
    plt.subplot(1, 2, 2)
    stat.probplot(df[feature], dist="norm", plot=plt)
```





```
# prompt: how to display heatmap
cor = df.corr()
import seaborn as sns
# Create a heatmap of the correlation matrix
sns.heatmap(data=cor, xticklabels=cor.columns.values, yticklabels=cor.columns.values)
plt.show()
```



```
# prompt: drop date month ,date day

df = df.drop('Date_month', axis=1)
df = df.drop('Date_day', axis=1)
```

```
len(df)
```

```
56420
```

```
len(df.columns)
```

```
23
```

```
df=data.sample(n=12000)
```

Dividing the dataset into Independent and Dependent features

```
# @title Dividing the dataset into Independent and Dependent features
x = df.drop(["RainTomorrow","Date"], axis=1)
y = df["RainTomorrow"]
```

```
names=x.columns
```

```
names
```

```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'Date_month', 'Date_day'],
      dtype='object')
```

```
from sklearn.preprocessing import StandardScaler # Make sure to import the class

sc = StandardScaler() # Instantiate the StandardScaler object with correct capitalization
x = sc.fit_transform(x)
```

```
ss = sc.feature_names_in_
```

```
x = pd.DataFrame(x, columns=names)
```

```
xc = x.columns
```

Train test split

```
# @title Train test split
from sklearn import model_selection
x_train,x_test,y_train,y_test = model_selection.train_test_split(x,y, test_size =0.2, random_state = 0)
```

```
y_train
```

```
78295     1
123659    0
45915     0
9119      0
34454     0
         ..
12501     0
132475    0
88787     0
102592    0
101131    1
Name: RainTomorrow, Length: 9600, dtype: int64
```

```
[ ] x_train.columns
```

```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'Date_month', 'Date_day'],
      dtype='object')
```

```
df.isnull().sum()
```

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
Date_month       0
Date_day         0
dtype: int64
```

Dividing the dataset into Independent and Dependent features

```
[ ] # @title Dividing the dataset into Independent and Dependent features
    x = df.drop(["RainTomorrow", "Date"], axis=1)
    y = df["RainTomorrow"]
```

Train test split

```
# @title Train test split
from sklearn import model_selection
x_train,x_test,y_train,y_test = model_selection.train_test_split(x,y, test_size =0.2, random_state = 0)
```

```
[ ] c = x_train.columns
```

```
[ ] len(x_train.columns)
```

```
23
```

```
[ ] names = ['Location', 'MinTemp', 'MaxTemp', 'Rainfall','Evaporation','Sunshine', 'WindGustDir',
             'WindGustSpeed','WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
             'Pressure9am', 'Pressure3pm', 'Cloud9am','Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday', 'Date_month', 'Date_day']
    len(names)
```

```
23
```

```
[ ] df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am
38275	2014-01-10	38	13.9	31.7	0.0	6.8	13.1	3	33.0	12	0	15.0	13.0	47.0	29.0	1020.3	1016.7	1.0	4.0	22.9
72016	2013-09-09	44	15.9	33.1	0.0	9.0	9.2	0	72.0	1	0	31.0	43.0	29.0	9.0	1012.7	1004.5	7.0	7.0	23.1
141207	2014-06-25	11	16.0	29.9	0.0	5.0	11.0	13	48.0	13	15	20.0	30.0	38.0	17.0	1018.1	1013.3	0.0	0.0	20.1

```
[ ] names=c
```

```
XGBoost = xgboost.XGBRFClassifier()
Rand_forest = sklearn.ensemble. RandomForestClassifier()
svm = sklearn.svm.SVC()
Dtree = sklearn.tree. DecisionTreeClassifier()
GBM = sklearn.ensemble.GradientBoostingClassifier()
log = sklearn.linear_model.LogisticRegression()
```

```
[ ] XGBoost.fit(x_train,y_train)
    Rand_forest.fit(x_train,y_train)
    svm.fit(x_train,y_train)
    Dtree.fit(x_train,y_train)
    GBM.fit(x_train, y_train)
    log.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
[ ] p1 = XGBoost.predict(x_train)
    p2 = Rand_forest.predict(x_train)
    p3 = svm.predict(x_train)
    p4 = Dtree.predict(x_train)
    p5 = GBM.predict(x_train)
    p6 = log.predict(x_train)
```

```
from sklearn import metrics
```

```
print("xgboost:",metrics.accuracy_score(y_train,p1))
print("rand_forestt:",metrics.accuracy_score(y_train,p1))
print("svm:",metrics.accuracy_score(y_train,p1))
print("Dtree:",metrics.accuracy_score(y_train,p1))
print("GBM:",metrics.accuracy_score(y_train,p1))
print("log:",metrics.accuracy_score(y_train,p1))
```

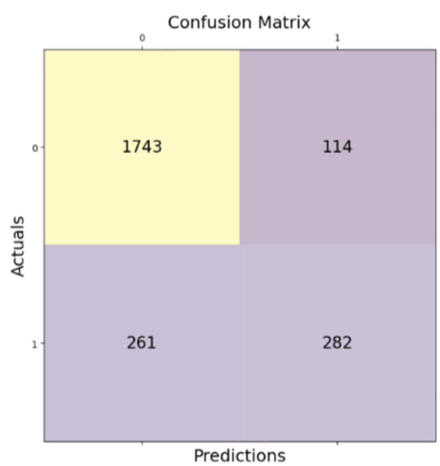
```
xgboost: 0.873125
rand_forestt: 0.873125
svm: 0.873125
Dtree: 0.873125
GBM: 0.873125
log: 0.873125
```

```
model = XGBoost
y_pred = model.predict(x_test)  # Generate predictions using the test set

conf_matrix = metrics.confusion_matrix(y_test, y_pred)
```

```
conf_matrix
```

```
array([[1743,  114],
       [ 261,  282]])
```

```
fig, ax = plt.subplots(figsize=(7.5, 7.5))
ax.matshow(conf_matrix, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```




```
import pickle
```

```
import pickle
```

```
import pickle
from sklearn.preprocessing import LabelEncoder

# Assuming you need a LabelEncoder, create one
le = LabelEncoder()
```

Double-click (or enter) to edit

```
# prompt: name 'imp_mode' is not defined

imp_mode = []
for i in range(len(x_train.columns)):
    imp_mode.append(x_train.iloc[:, i].mode()[0])
```

```
pickle.dump(model,open('rainfall.pkl1','wb'))
pickle.dump(sc,open('scale.pkl1','wb'))
```