Developing Machine Learning for Rainfall Prediction Using Random Forest: A Comparative Analysis of Classification Algorithms

Machine learning mini project

Kalasalingam University, February 2025

Presented by:

ARYA S NAIR - 9224146031@cdoe.klu.ac.in

VISHNU K G - 9224146108@cdoe.klu.ac.in

Table of contents:

For Data Analysis implementation,

https://github.com/arootty24/rainfall-prediction-ML.git

1. Abstract

Data mining is a process that aims to extract useful knowledge from cluttered and unorganized information. Climate change is a discipline involved with analyzing the varying distribution of weather for a specific period of time. Specifically, rainfall forecasting analyzes specific features such as humidity and wind are used to predict rainfall in specific locations. Rainfall prediction has of recent been subjected to several machine learning techniques with different degree of short-term (daily) and long-term (monthly) prediction performance. Selecting an appropriate technique for specific rainfall duration is a challenging task. Several approaches have been proposed for rainfall forecasting using various machine techniques. Three features—relative humidity, temperature, and wind direction—are the data that are employed. This study aims to provide a comparative analysis of the multiple machine learning classifiers for rainfall prediction based on random forest data.

KEYWORDS: Rainfall Prediction, Machine Learning

2. Introduction

Weather forecasting is a task which combines science and technology to predict the state of the atmosphere for a future time and a given location. Human has long attempted to predict the weather since ancient times. One of the main fields of weather forecasting is rainfall prediction, which is important for food production plan, water resource management and all activity plans in the nature. The occurrence of prolonged dry period or heavy rain at the critical stages of the crop growth and development may lead to significantly reduce crop yield, and rainfall prediction is a key tool for human survivability.

Random Forest algorithms are effective for rainfall prediction because they can handle complex, non-linear relationships in meteorological data. By constructing an ensemble of decision trees, Random Forests mitigate overfitting and improve prediction accuracy. The algorithm's capacity to assess feature importance also provides valuable insights into the key factors influencing rainfall patterns.

The presence of water, which comes from rain, is essential to the existence of all living things. When other water sources, like rivers, lakes, or wells, are unreachable, rain is a critical supply of water. Rainwater offers a host of advantages, for instance to irrigate agricultural land, industries, and power plants. Depending on factors including topography, wind patterns, climate, and geographic position, rainfall characteristics might differ between places. On the other hand, heavy rains can trigger landslides and floods. Although the amount of rainfall in the future cannot be predicted with precision, it can be calculated by utilizing historical and current weather data.

Predicting when it will rain is crucial for many industries, including agriculture, urban planning, water resource management, and disaster mitigation. One promising way to improve rain forecasting accuracy and dependability is to employ machine learning algorithms. Using machine learning, an advanced computational technique, one can

leverage the potential of large datasets to find complex relationships, trends, and patterns across different meteorological variables. Machine learning algorithms can better comprehend and anticipate rainfall patterns by using past data to generate precise predictions. Basically, rain prediction is the process of estimating the probability of future rain events by examining current and historical weather data along with other environmental conditions. Prediction models can make more accurate forecasts by taking advantage of intricate patterns and correlations between weather variables using machine learning techniques.

This research focused on random forest algorithms to create and evaluate machine learning-based rain prediction models, random forests algorithm is a popular ensemble learning method used for classification. It is anticipated that this research will advance knowledge of the efficiency of machine learning algorithms in rain prediction and offer direction for the creation of more advanced and dependable rain prediction systems. Ultimately, it is hoped that the use of machine learning techniques in rain prediction can help society and the government in responding appropriately to the effects of extreme weather, lower financial losses, and boost resilience to natural disasters.

3. Existing System Study

Existing systems for rainfall prediction using machine learning typically leverage a range of algorithms, with Random Forest being a popular choice due to its ability to handle large, complex datasets and produce accurate predictions. These systems generally involve the collection of meteorological data such as temperature, humidity, and wind speed, which are then processed and used to train the model. Random Forest, as an ensemble learning method, constructs multiple decision trees and aggregates their outputs, making it robust to overfitting and effective in capturing non-linear relationships within the data. Comparative studies often evaluate the performance of Random Forest against other classification algorithms in terms of accuracy, precision, recall, and computational efficiency. These systems aim to optimize prediction accuracy while minimizing errors like false positives or negatives, and address challenges like feature selection and model interpretability, offering valuable insights for improving weather forecasting and decision-making processes in agriculture and disaster management.

4. Drawbacks

The existing systems for rainfall prediction using machine learning, particularly with Random Forest, have several drawbacks. One major issue is the complexity of feature selection and preprocessing, as raw meteorological data often requires significant cleaning and transformation to be useful for model training. Random Forest models can also suffer from overfitting, especially when the dataset is small or the model is not properly tuned, leading to poor generalization on unseen data. Additionally, while Random Forest is an effective ensemble method, it lacks transparency and interpretability, making it difficult for users to understand how predictions are made, which is crucial in meteorological applications. Moreover, comparative studies of Random Forest with other classification algorithms can be inconclusive, as algorithm performance may heavily depend on the specific characteristics of the dataset, the quality of feature engineering, and the hyperparameter tuning process, which can be computationally intensive and time-consuming. These limitations make it challenging to deploy these systems in real-time forecasting scenarios with high confidence.

5. Problem Statement

The problem is to develop an accurate rainfall prediction model using the Random Forest algorithm, leveraging historical weather data. The challenge lies in optimizing the model for generalization and addressing issues related to overfitting and interpretability.

6. Objectives

1. To develop a robust machine learning model for predicting rainfall based on historical meteorological data, such as temperature, humidity, wind speed, and pressure.

2. To evaluate the performance of the Random Forest algorithm by comparing it with other classification algorithms in terms of accuracy, precision, recall, and computational efficiency.

3. To optimize the Random Forest model by tuning hyperparameters and addressing issues like overfitting to enhance generalization on unseen data.

4. To improve model interpretability by analyzing feature importance, helping meteorologists understand the key factors influencing rainfall predictions.

5. To provide a practical and reliable tool for real-time rainfall forecasting, supporting decision-making in sectors like agriculture, disaster management, and water resource management.

7. Methodology

7.1 Dataset Used:

The dataset used in this study is the rainfall prediction at random forest , which contains 366 entries with 12 features, including day , pressure, humidity and etc .

	day	pressure	maxtemp	temparature	mintemp	dewpoint	humidity	cloud	rainfall	sunshine	winddirection	windspeed
0	1	1025.9	19.9	18.3	16.8	13.1	72	49	yes	9.3	80.0	26.3
1	2	1022.0	21.7	18.9	17.2	15.6	81	83	yes	0.6	50.0	15.3
2	3	1019.7	20.3	19.3	18.0	18.4	95	91	yes	0.0	40.0	14.2
3	4	1018.9	22.3	20.6	19.1	18.8	90	88	yes	1.0	50.0	16.9
4	5	1015.9	21.3	20.7	20.2	19.9	95	81	yes	0.0	40.0	13.7

1.1 Rainfall Dataset

	pressure	maxtemp	temparature	mintemp	dewpoint	humidity	cloud	rainfall	sunshine	winddirection	windspeed
count	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000
mean	1013.742623	26.191257	23.747268	21.894536	19.989071	80.177596	71.128415	0.680328	4.419399	101.284153	21.534153
std	6.414776	5.978343	5.632813	5.594153	5.997021	10.062470	21.798012	0.466988	3.934398	81.722827	10.056054
min	998.500000	7.100000	4.900000	3.100000	-0.400000	36.000000	0.000000	0.000000	0.000000	10.000000	4.400000
25%	1008.500000	21.200000	18.825000	17.125000	16.125000	75.000000	58.000000	0.000000	0.500000	40.000000	13.725000
50%	1013.000000	27.750000	25.450000	23.700000	21.950000	80.500000	80.000000	1.000000	3.500000	70.000000	20.500000
75%	1018.100000	31.200000	28.600000	26.575000	25.000000	87.000000	88.000000	1.000000	8.200000	190.000000	27.825000
max	1034.600000	36.300000	32.400000	30.000000	26.700000	98.000000	100.000000	1.000000	12.100000	350.000000	59.500000

1.2 Descriptive Statistics

7.2 Data Preprocessing

The DataFrame contains 366 entries (presumably daily data for a year) and 12 columns, each representing a different feature. Most columns, such as 'day', 'pressure', 'maxtemp', 'temparature', 'mintemp', 'dewpoint', 'humidity', 'cloud', 'sunshine', 'winddirection', and 'windspeed'. However, the 'rainfall' column is of type object, suggesting it might contain non-numeric data or a mix of data types, which will require specific preprocessing. Notably, 'winddirection' and 'windspeed' have one missing value each (365 non-null entries instead of 366), indicating the need for imputation or removal of these missing values during preprocessing.

This included identifying null entries in key variables such as temperature and rainfall , which were later filled with median and mode values to preserve the dataset's completeness. Additionally, outliers were examined to ensure they did not skew model predictions. Numerical scaling and normalization techniques were considered to standardize features, allowing the models to train efficiently. By carefully preprocessing the data, we established a strong foundation for subsequent feature engineering and model building.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   day                     366 non-null    int64
 1   pressure                366 non-null    float64
 2   maxtemp                 366 non-null    float64
 3   temparature             366 non-null    float64
 4   mintemp                 366 non-null    float64
 5   dewpoint                366 non-null    float64
 6   humidity                366 non-null    int64
 7   cloud                   366 non-null    int64
 8   rainfall                366 non-null    object
 9   sunshine                366 non-null    float64
 10          winddirection   365 non-null    float64
 11  windspeed               365 non-null    float64
dtypes: float64(8), int64(3), object(1)
memory usage: 34.4+ KB
```

1.3 Dataset info

Data Preprocessing

```
# drop highly correlated column
data = data.drop(columns=['maxtemp', 'temparature', 'mintemp'])
```

```
data.head()
```

	pressure	dewpoint	humidity	cloud	rainfall	sunshine	winddirection	windspeed
0	1025.9	13.1	72	49	1	9.3	80.0	26.3
1	1022.0	15.6	81	83	1	0.6	50.0	15.3
2	1019.7	18.4	95	91	1	0.0	40.0	14.2
3	1018.9	18.8	90	88	1	1.0	50.0	16.9
4	1015.9	19.9	95	81	1	0.0	40.0	13.7

```
#upsampling data using Smote
```

```
X=data.drop(columns=["rainfall"],axis=1)
y=data.rainfall
```

```
#Over sampling the minority data using smote
smote=SMOTE(sampling_strategy='minority')
X_upsample,y_upsample=smote.fit_resample(X,y)
```

```
y_upsample.value_counts()
```

```
rainfall
1    249
0    249
Name: count, dtype: int64
```

```
# splitting the data into training data and test data
X_train, X_test, y_train, y_test = train_test_split(X_upsample, y_upsample, test_size=0.2, random_state=42)
```

1.4 Data Preprocessing

8. Data Analysis

The analysis focuses on understanding the underlying patterns and characteristics of the selected numerical features. Histograms provide a visual representation of the frequency distribution, revealing whether the data is normally distributed, skewed, or multimodal. KDEs offer a smoothed estimate of the probability density function, highlighting the overall shape of the distribution and potential peaks or valleys. By examining these plots, one can identify features with unusual distributions, such as those with significant skewness or multiple modes, which might require further investigation or preprocessing. For instance, highly skewed features might benefit from transformations to normalize their distribution, while multimodal features could indicate the presence of distinct subgroups within the data. Additionally, the presence of outliers can be identified, which might need to be addressed depending on their impact on the analysis or modeling. This analysis helps in gaining a deeper understanding of the data's structure and informing subsequent steps in the data preprocessing and modeling pipeline.

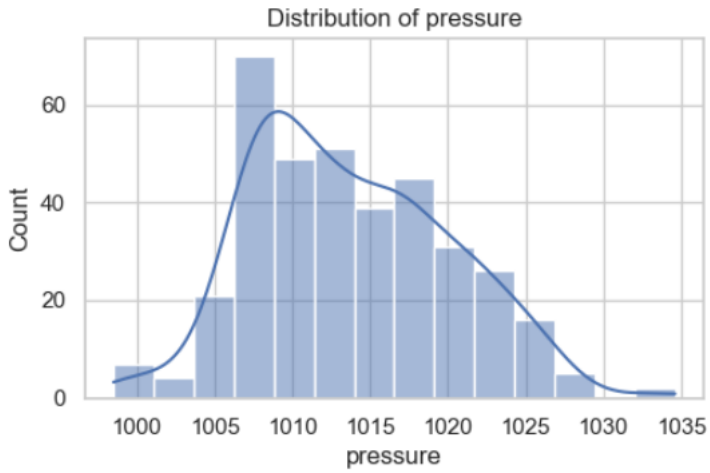
```
plt.figure(figsize=(15, 10))

for i, column in enumerate(['pressure', 'maxtemp', 'temparature', 'mintemp', 'dewpoint', 'humidity','cloud', 'sunshine', 'windspeed'], 1):
  plt.subplot(3, 3, i)
  sns.histplot(data[column], kde=True)
  plt.title(f"Distribution of {column}")

plt.tight_layout()
plt.show()
```

1.5 Data Analysis

9. Distribution of Data Collections

9.1 Distribution of pressure

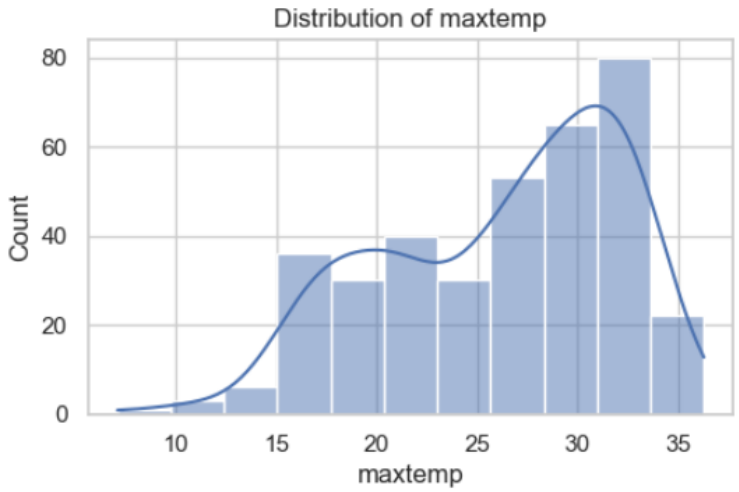
This image shows the distribution of atmospheric pressure, displaying a histogram with a superimposed kernel density estimate. The distribution appears roughly bell-shaped, suggesting a tendency towards a normal distribution with a slight skew towards higher pressures.



1.6 Distribution of pressure

9.2 Distribution of Maxtemp

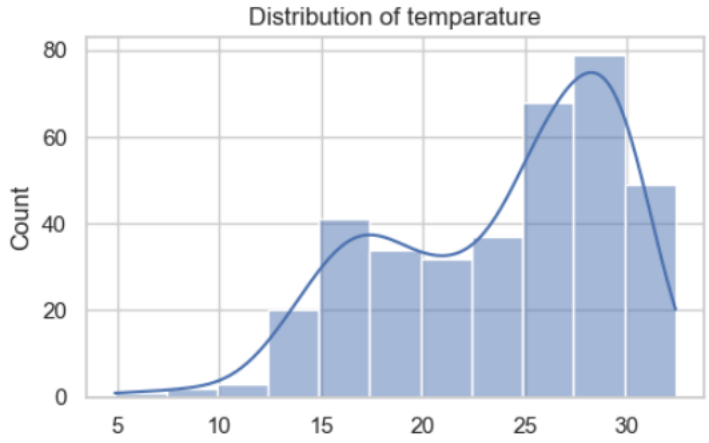
This graph shows how often different maximum temperatures ("maxtemp") occur. It looks like the most common maximum temperatures are between 30 and 35, with a smaller peak around 15 to 20.



1.7 Distribution of Maxtemp

9.3 Distribution of Temperature

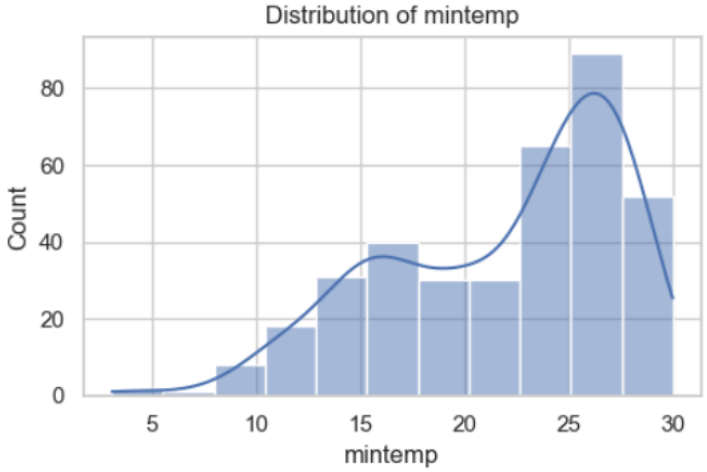
This graph shows how often different temperatures ("temparature") occur. Most temperatures are between 25 and 30, with a smaller group around 15 to 20.



1.8 Distribution of Temperature

9.4 Distribution of Mintemp

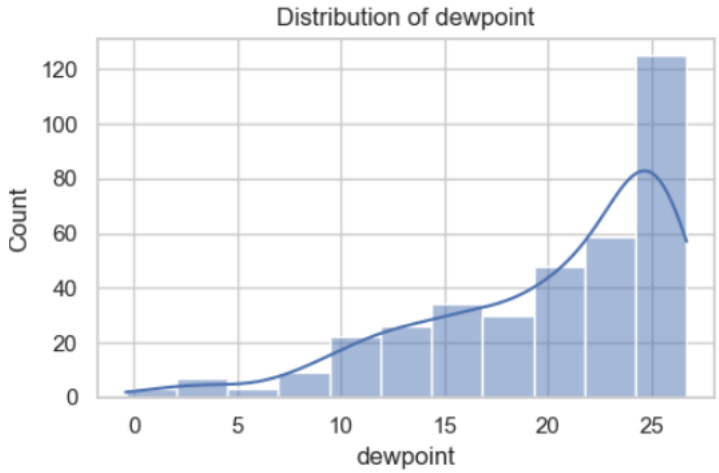
The graph illustrates the frequency of different minimum temperatures ("mintemp"). It reveals that the most common minimum temperatures fall between 25 and 30 degrees, while a smaller cluster of occurrences is observed in the 15 to 20 degree range.



1.9 Distribution of pressure

9.5 Distribution of Dewpoint

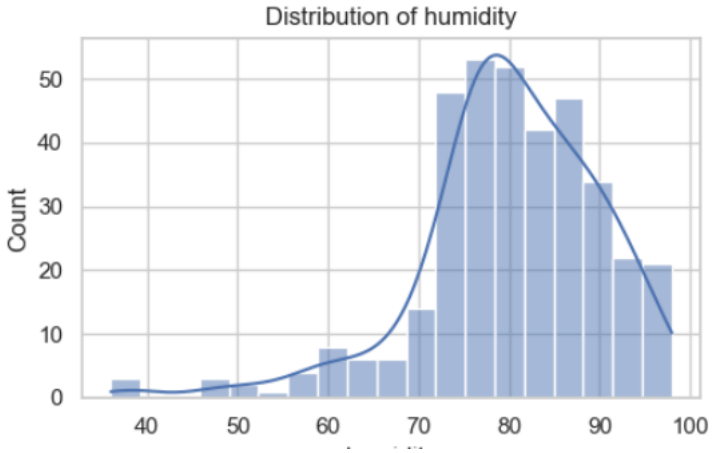
Dew points around 25 are most common, with fewer occurrences at lower values, indicating higher moisture is more frequent.



1.10 Distribution of Dewpoint

9.6 Distribution of Humidity

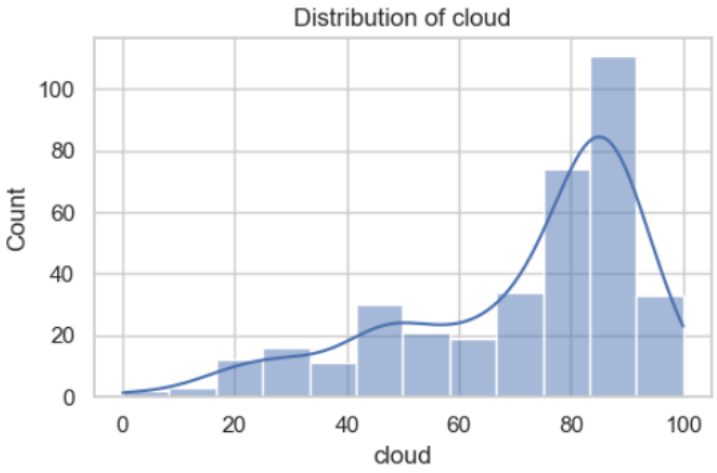
This graph shows the distribution of humidity levels. The most frequent humidity levels are between 75% and 90%, with a peak around 80%. Humidity levels below 60% and above 95% are relatively rare.



1.11 Distribution of humidity

9.7 Distribution of Cloud

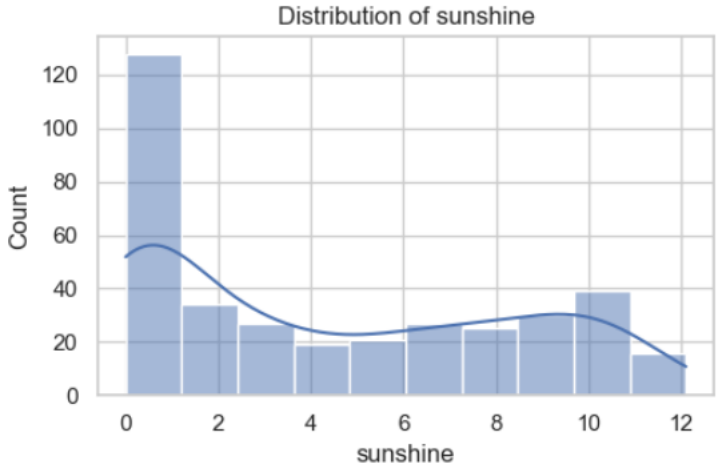
This graph shows the distribution of cloud cover, measured as a percentage. It indicates that high cloud cover (around 80-100%) is the most frequent, with a secondary peak around 40-60%. Low cloud cover (0-20%) is relatively uncommon.



1.12 Distribution of Cloud

9.8 Distribution of Sunshine

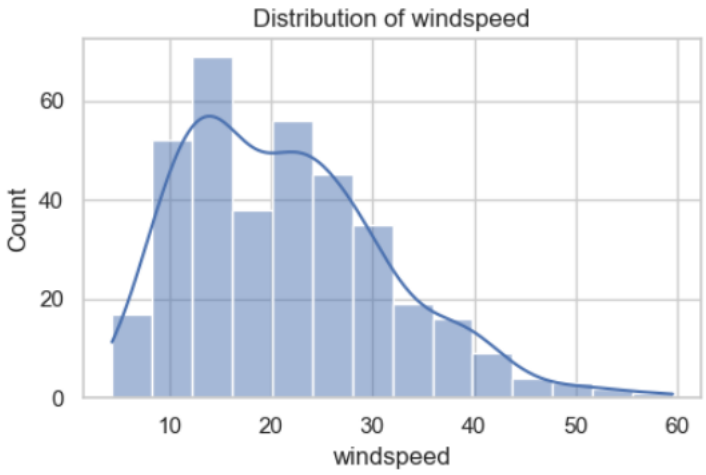
This graph shows the distribution of sunshine hours. It reveals that the most frequent occurrence is for very low sunshine hours, close to zero. There's a secondary peak around 10-12 hours of sunshine, indicating a bimodal distribution. Sunshine hours between 2 and 10 are relatively less common.



1.13 Distribution of Sunshine

9.9 Distribution of Windspeed

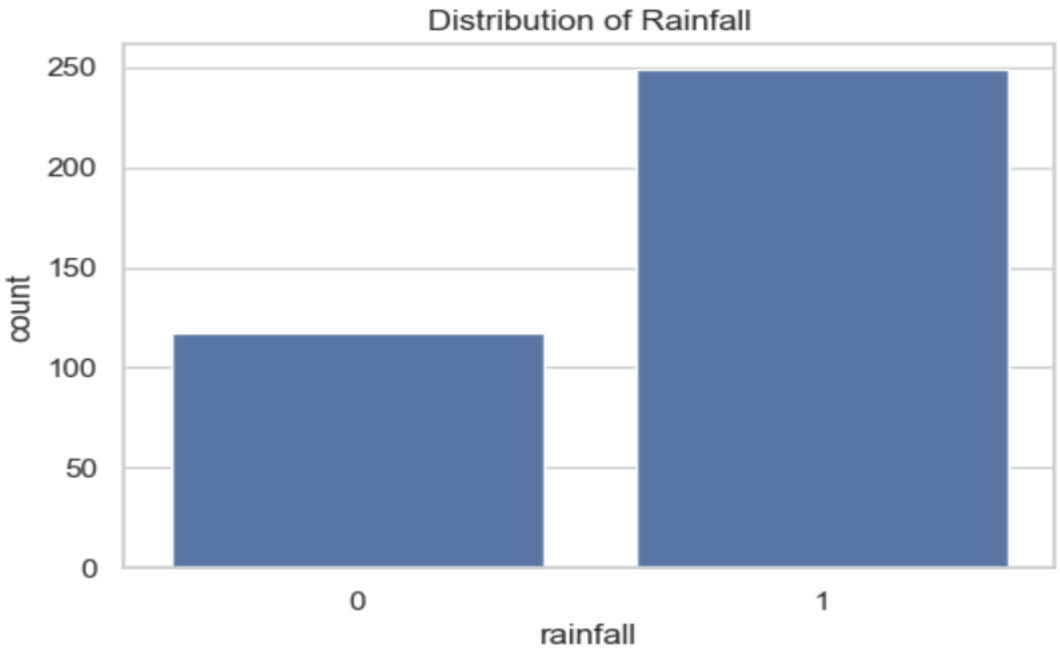
This graph displays the distribution of wind speeds. It shows that wind speeds are most frequently between 10 and 20, with a peak around 15. Higher wind speeds become progressively less common, with very few instances above 40.



1.14 Distribution of Windspeed

10. Rainfall Frequency Distribution

This graph depicts the distribution of rainfall in a simplified binary format, where "0" likely represents no rainfall and "1" indicates the presence of rain. A striking imbalance is evident, with the "rain" category ("1") significantly outnumbering the "no rain" category ("0"). Specifically, there are over 200 instances of rain recorded, compared to just over 100 instances of no rain. This skew towards rainy days suggests a potential bias in the dataset that could affect the performance of a predictive model. It's crucial to understand the specific definitions of "0" and "1" in this context to fully grasp the implications of this distribution and to consider appropriate techniques to mitigate any potential bias during model training.

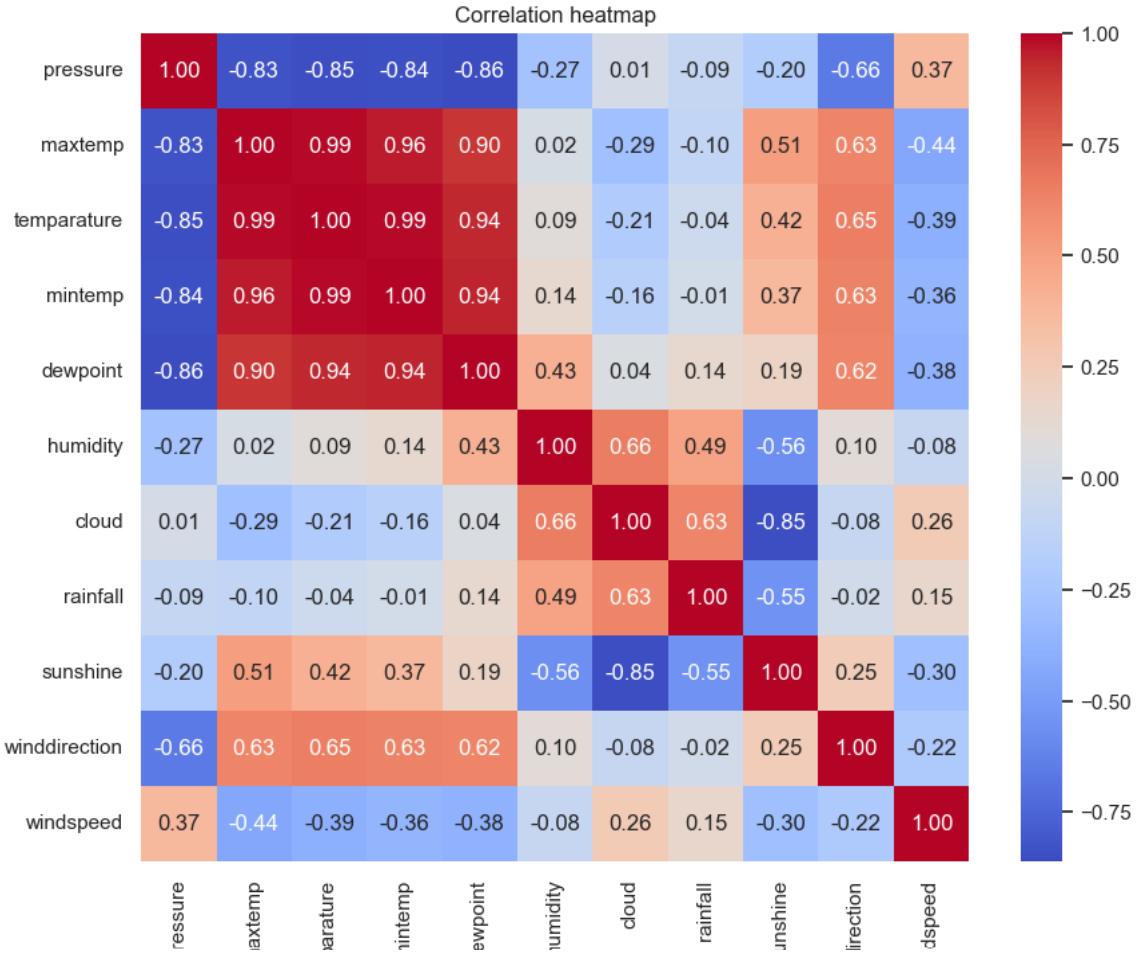


1.15 Distribution of Rainfall

11. Correlation Analysis And Result

This image displays a correlation heatmap, which visually represents the relationships between different weather variables. Each cell in the grid shows the correlation coefficient, a number between -1 and 1, indicating how strongly two variables are related. Red colors indicate a strong positive correlation (variables increase together), blue colors indicate a strong negative correlation (one variable increases as the other decreases), and white or light colors indicate a weak or no correlation.The heatmap allows us to quickly identify patterns and understand how different weather factors interact. For instance, we can see that temperature-related variables ("maxtemp", "temparature", "mintemp", "dewpoint") are highly positively correlated with each other and negatively correlated with "pressure".

This suggests that when temperatures rise, air pressure tends to drop. Additionally, "humidity" shows a moderate positive correlation with "cloud" and "rainfall", implying that higher humidity is often associated with more clouds and rain. The heatmap provides a valuable overview for understanding the complex relationships between these variables and can be crucial for building accurate weather prediction models.



1.16 Correlation of Heatmap

12. Model Parameters and Selection

This code defines a parameter grid for hyperparameter tuning of a Random Forest Classifier model. The param_grid_rf dictionary specifies the range of values to be explored for each hyperparameter. n_estimators controls the number of trees in the forest, with options of 50, 100, or 200. max_features determines the number of features to consider when looking for the best split, using either the square root ("sqrt") or logarithm base 2 ("log2") of the total features. max_depth sets the maximum depth of each tree, with options including no limit (None) or specific depths (10, 20, 30). min_samples_split specifies the minimum number of samples required to split an internal node, with values of 2, 5, or 10. min_samples_leaf sets the minimum number of samples required to be at a leaf node, with options of 1, 2, or 4. These parameters will be used in a grid search or randomized search to find the optimal combination that maximizes the model's performance.

```
# Define the parameter grid for hyperparameter tuning
rf_model = RandomForestClassifier(random_state=42)

param_grid_rf = {
    "n_estimators": [50, 100, 200],
    "max_features": ["sqrt", "log2"],
    "max_depth": [None, 10, 20, 30],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4]
}
```

1.17 Hyperparameter tuning

13. Advantages of Proposed Approach

The proposed approach, employing a parameter grid for hyperparameter tuning of a Random Forest Classifier, offers significant advantages in model optimization. This method aims to identify the configuration that maximizes the model's performance on the given dataset. This process not only leads to improved accuracy and generalization by mitigating overfitting, but also ensures a tailored model that aligns with the specific characteristics of the data. Furthermore, the explicit definition of the parameter grid ensures reproducibility, allowing for consistent results and transparent model development. Ultimately, this approach enhances the model's robustness and predictive capabilities, crucial for reliable and effective machine learning applications.

14. Conclusion

This code prepares a Random Forest model for optimized training. It initializes a classifier with a fixed random seed for consistency. A parameter grid is then created to systematically test various hyperparameter combinations. This grid includes options for tree count, feature selection, tree depth, and sample requirements. The goal is to find the best settings for the model's structure. By using this grid in a search, performance will be maximized. Ultimately, this leads to a more accurate and reliable Random Forest model. This approach demonstrates a key technique for improving machine learning models.

15. Future Enhancements

- **Feature Engineering:**

Investigate and create new features from existing data that might improve the model's predictive power. Apply feature selection techniques to identify and use only the most relevant features, reducing noise and improving efficiency.

- **Model Evaluation and Validation:**

Evaluate the model using a wider range of performance metrics, beyond just accuracy, to gain a more comprehensive understanding of its strengths and weaknesses. Implement learning curves to understand if more data would help the model.

- **Model Explainability:**

Use techniques like feature importance analysis or SHAP values to understand which features are most influential in the model's predictions, improving model interpretability.